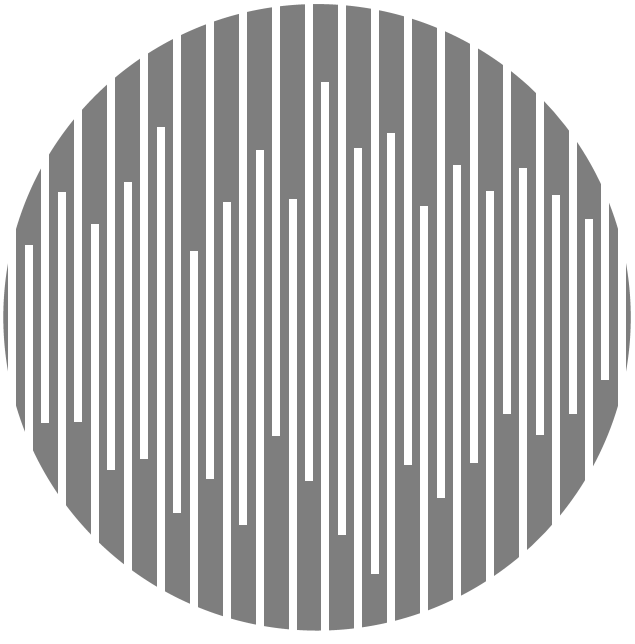
Modicon Micro Ladder Logic Benutzerhandbuch

A91M.12-702311.21-0899



GROUPE SCHNEIDER

■ Merlin Gerin ■ Modicon ■ Square D ■ Telemecanique

Hinweise

Anwendungshinweis

Achtung: Für Anwendungen bei Steuerungen mit sicherheitstechnischen Anforderungen sind die einschlägigen Vorschriften zu beachten.
Reparaturen an Komponenten sollen aus Gründen der Sicherheit und Erhaltung der dokumentierten Systemdaten nur durch den Hersteller erfolgen.

Schulung

Zur Vermittlung ergänzender Systemkenntnisse werden von AEG entsprechende Schulungen angeboten (siehe Anschriften).

Daten, Abbildungen, Änderungen

Daten und Abbildungen sind unverbindlich. Änderungen, die dem technischen Fortschritt dienen, sind vorbehalten. Falls Sie Verbesserungs- oder Änderungsvorschläge haben oder Fehler in dieser Druckschrift entdecken sollten, bitten wir um Ihre Mitteilung.

Copyright

Kein Teil dieser Dokumentation darf ohne schriftliche Genehmigung der AEG Aktiengesellschaft in irgendeiner Form reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Die Übersetzung in eine fremde Sprache ist nicht gestattet.

Warenzeichen

Die in diesem Handbuch für die AEG Produkte verwendeten Bezeichnungen sind im allgemeinen Warenzeichen der AEG Aktiengesellschaft.

© 1993 AEG Aktiengesellschaft

Inhalt

Kapitel 1 Kontaktplanlogik- Betriebssystem für die Modicon Micro SPS 1

Die speicherprogrammierbaren Steuerungen der Modicon Micro Serie 2
Theoretischer Hintergrund der Arbeitsweise 2
Eingänge 2
CPU 4
Flash-Speicher 4
Ausgänge 4
Systemleistung 4
Zykluszeit 4
Logik-Bearbeitungszeit 4
Speicherverwaltung 5
Anwender-Datenspeicher 5
Numerierung der Referenzen 5
System-Konfigurationsspeicher 6
Anwender-Programmspeicher 7
Speicher-Pufferung 8
Optionale Pufferung-Verfahren 8
Pufferung mit Hilfe des Flash-Speichers 8
SPS Einschalt-Prüfung 8
Lagerung einer SPS mit im Flash gespeicherter Anwenderlogik . . . 9
SPS Betriebsarten 10
Die E/A-Erweiterungskopplung 10
A120 E/A-Erweiterung 12
Kontaktplanlogik-Befehlsvorrat 14

Kapitel 2 Start-Prozeduren 17

Inbetriebnahme 18
Einschalten der Stromversorgung 18
Starten einer bereits konfigurierten SPS 18
Das Starten einer unkonfigurierten SPS 19
Konfigurierung einer Modicon Micro SPS 19

Parameter der automatischen Konfiguration 20
Autokonfiguration einer SPS in der Betriebsart Single 20
Autokonfiguration einer SPS in der Betriebsart Parent 21
Autokonfiguration einer SPS in der Betriebsart Child 22
Einige Beispiele für Autokonfigurationen 23
SPS Betriebsart 24
Child-ID # 24
0x, 1x, 3x und 4x Referenzbereiche 24
Anzahl der Kontaktplanlogik-Segmente 24
Anzahl des Child-Steuerungen 25
E/A-Bereiche 25
Die Batteriespule 25
Das Timer-Register 25
Die Uhr 26
Automatisch konfigurierte Kommunikations-Ports 27
Der RS-485 Port 27
Die RS-232 Ports 27
Modifizierung der Konfigurationsparameter 29
Die Anzahl der Referenzen 29
Anzahl der Logiksegmente 30
RS-232 Kommunikationsparameter des RS-232 Port 30
Modbus-Modus 30
Einfacher ASCII-Modus 31
Modem Kommunikationsmöglichkeiten 31
Kommunikationsparameter des RS-485 Port 31
Adressierung der E/A-Bereiche 32
Feste E/A-Bereiche 32
Adressierung der A120 E/A 34
Beispiel: Eine Micro SPS mit einem A120 E/A-Baugruppenträger 34
Adressierung der E/A in einer Erweiterungskopplung 36
Die Parent-SPS 36
Eine Child-SPS 37
Beispiel: Eine Erweiterungskopplung, bei der die gesamte, lokale E/A vom Parent gesteuert wird 37
Aufteilung lokaler E/A zwischen Parent- und den Child-SPS 39
Beispiel: Aufgeteilte E/A 40
Generalisierter Datentransfer 42
SPS Operationen 44

Kapitel 3 Grundlagen der Kontaktplanlogik-Programmierung 45

Segmente und Netzwerke 46
Kontaktplanlogik-Segmente 46

Kontaktplan-Netzwerke 46
Einsetzen von Schaltlogik und Befehlen in einem Netzwerk 46
Wie Kontaktplanlogik abgearbeitet wird 47
Schaltlogikelemente 48
Schaltkontakte 48
Normale und speicherresidente Spulen 49
Darstellung von Spulen in einem Netzwerk 49
Vertikale und Horizontale Verbindungen 49
Anwendungsbeispiel: Start/Stop-Schaltung für einen Motor 51

Kapitel 4 Zähler und Timer 53
Zählerbefehle 54
Beispiel für einen einfachen Vorwärtzähler 54
Timerbefehle 55
Beispiel eines Timers für eine Sekunde 55
Anwendungsbeispiel: Eine Echtzeit-Uhr mit einem Millisekunden-Timer 56

Kapitel 5 Grundlegende Mathematikbefehle 57
Integer-Mathematikbefehle 58
Ein DIV Beispiel 59
Anwendungsbeispiel: Umwandlung von Fahrenheit in Celsius 60

Kapitel 6 Datenmanagement-Befehle 61
Verschieben von Register- und Tabellendaten 62
Anlage eines FIFO Stack 64
Suche in einer Tabelle 66
Beispiel für eine SRCH Operation 67
Verschieben eines Datenblocks 68
Anwendungsbeispiel: Routine zum Laden eines Rezepts mithilfe Blockverschiebung 68

Kapitel 7 Datenmanipulations-Befehle 71
Befehle für Boolesche Logik 72
ODER 74
Exklusives ODER 75
Archivierung der ursprünglichen Zielmatrixwerte 75
Ein Anwendungsbeispiel: einfache Berechnung des Tabellendurchschnitts 76
Bit-Komplementbildung in einer Datenmatrix 77
Beispiel eines Bit-Komplements 78
Bit-Vergleich in einer Datenmatrix 79
Beispiel eines Bit-Vergleichs 80
Bit-Abfrage und -Veränderung in einer Datenmatrix 81

Kapitel 8 Einfache ASCII Kommunikation 85
ASCII-Kommunikation mit Hilfe der Kontaktplanlogik 86
Definierte Nachrichtenformate 86
Der COMM-Befehl 88
Datenformate 92
ASCII Zeichenformat 92
Integerformat (1 ... 4) 93
Hex-Format (1 ... 4) 93
Flush (Löschen) des Eingabepuffer-Formats 94
Flush (Löschen) des Eingabebyte-Formats 94
ASCII Zeichensatz 95
Anwendungsbeispiel: Benutzung des HHP als ASCII Datenanzeigegerät 97

Kapitel 9 Die Schnittstelle zur Reihenfolgensteuerung 101
SCIF-Befehl 102
Anwendungsbeispiel: Zeit-Schrittgeber mit SCIF-Blöcken 106
Referenzmarkierungen für das Anwendungsbeispiel 108

Kapitel 10 Befehle für Unterprogramme 113
Befehle für Unterprogramme in der Kontaktplanlogik 114
Die Interrupt- und Zähler-/Timer-Eingänge 116
Hardware-Interrupt Operation 116
Der Hochgeschwindigkeitszähler-Eingang 117
Der CTIF Befehl 118
Ein CTIF Anwendungsbeispiel 121
Segment 1, Netzwerk 1 121
Segment 1, Netzwerk 2 125
Die Unterprogramme von Segment 2 126

Kapitel 11 Andere Standard-Befehle 127
Das Überspringen von Netzwerken 128
Ein einfaches SKP-Beispiel 128
Prüfung des Status der SPS 129
Die Statustabelle der Modicon Micro SPS 130
Statustabelle der Modicon Micro SPS (Fortsetzung) 131
Statustabelle der Modicon Micro SPS (Fortsetzung) 132
Statustabelle der Modicon Micro SPS (Fortsetzung) 133
Statustabelle der Modicon Micro SPS (Fortsetzung) 134
Statustabelle der Modicon Micro SPS (Fortsetzung) 135
Zykluszeit-Funktionen 136

Kapitel 12 Erweiterter Befehlssatz verfügbar auf ausgewählten Micro SPS Modellen 137
BlockeTabelle Verschiebungsbefehle 138
Der Checksum-Befehl 139
Der Proportional-Integral-Differerential-Befehl 140
Erweiterte mathematische Befehle 145
Erweiterte mathematische Befehle (Fortsetzung) 150

Anhang A Update des Betriebssystems im Flash 157
Dienstprogramme für Betriebssystem-Updates 158
Update des Systems mit einem Kontaktplanlogik-Programm 158
Feststellen der neuesten, verfügbaren Version 159
Zugang zu Modfax 160
Zugang zum Customer Service Bulletin Board 160
Betriebssystem Update-Prozedur 161

Anhang B Fehlerbeseitigung 163
Diagnose der Startbedingungen 164
Symptom: powerok LED leuchtet nicht 164
Symptom: readyLED leuchtet nicht 164
Symptom: run LED leuchtet nicht oder blinkt 165
Symptom: exp link LED leuchtet nicht oder blinkt 165
Symptom: Keine Kommunikation zur SPS 165
SPS Stopfehler-Codes 168
An den LEDs angezeigte SPS-Crashcodes 169

Kapitel 1 Kontaktplanlogik-Betriebssystem für die Modicon Micro SPS

- ☐ Die speichereprogrammierbaren Steuerungen der Modicon Micro Serie
- ☐ Speicherverwaltung
- ☐ Speicher-Pufferung
- ☐ Wahl einer SPS-Betriebsart
- ☐ Das Kontaktplanlogik-Befehlsvorrat

Die speicherprogrammierbaren Steuerungen der Modicon Micro Serie

Eine speicherprogrammierbare Steuerung (SPS) ist ein Gerät auf der Basis digitaler Datenverarbeitung, das für die Echtzeit-Steuerung in Industrie und Herstellung entwickelt wurde. Eine SPS umfaßt Eingabe- und Ausgabe-Einheiten (E/A) und eine zentrale Prozessor-Einheit (CPU).

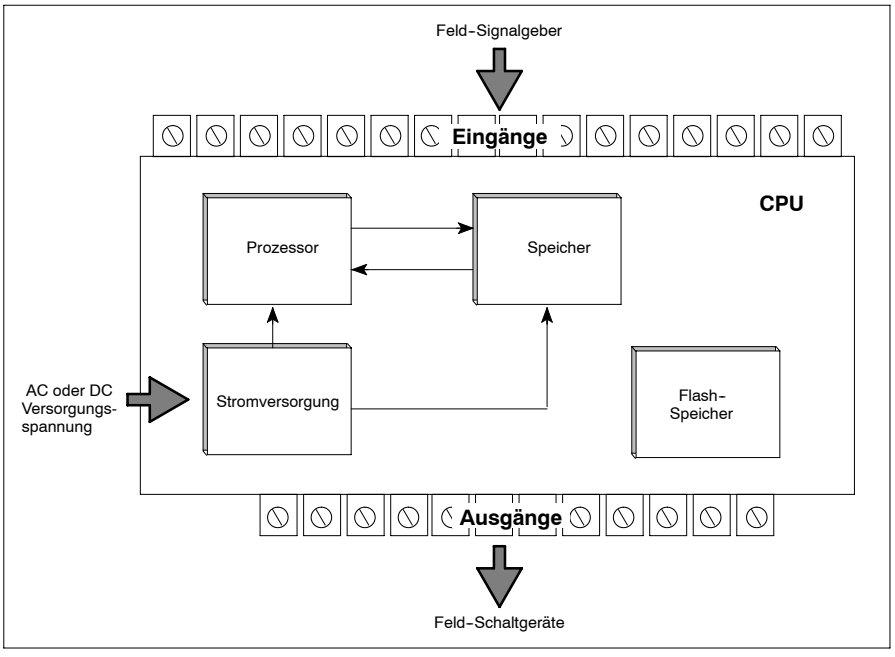
Modicon Micro-SPS haben fest eingebaute E/A-Module. Die Ein- und Ausgabekomponenten bilden zusammen mit der CPU eine gemeinsame Baueinheit. Aufgrund dieser Bauweise ergibt sich ein kleines, leichtes, kostengünstiges und komplettes Paket, das Lösungen für ein breites Spektrum von Steuerungsanwendungen bietet.

Theoretischer Hintergrund der Arbeitsweise

Das untenstehende Blockdiagramm zeigt die wichtigsten Komponenten einer Micro-SPS. Die SPS überwacht den Zustand der Feldgeräte, indem sie Signale über die Eingänge erhält, ein in der CPU gespeichertes, vom Benutzer eingegebenes Logikprogramm bearbeitet, und dann auf dieser Grundlage die Aktivität anderer Geräte im Feld steuert, indem zu diesen Geräten Signale über die Ausgänge gesendet werden.

Eingänge

Die Eingänge befinden sich in einem Anschlußblock auf der Oberseite der SPS. Sie werden im Feld an die Sensorelemente Ihrer Anwendung angeschlossen, also z.B. an Taster, Auswahlschalter, Motoreinschaltkontakte, Drehschalter oder Begrenzungsschalter. Wenn ein Eingang feststellt, daß ein Schalter geschlossen wird, wandelt dieser Eingang das Spannungssignal in ein Logiksignal um, das diesen Zustand des Sensors beschreibt und von der CPU ”verstanden” werden kann eine logische 1 bezeichnet den Zustand EIN oder GESCHLOSSEN, eine logische 0 den Zustand AUS oder OFFEN.


Feld-Signalgeber
Eingänge
CPU
Prozessor
Speicher
AC oder DC
Versorgungs-
spannung
Stromversorgung
Flash-
Speicher
Ausgänge
Feld-Schaltgeräte

CPU
Innerhalb der CPU befinden sich der digitale Prozessor, der Speicher und die Stromversorgung. Diese Komponenten arbeiten zusammen, um das Logikprogramm der Anwendung abzuarbeiten und Steuersignale an die Ausgänge zu schicken. Die CPU liest die umgewandelten Eingangsdaten, führt das im Speicher befindliche Logikprogramm aus, und schreibt die entsprechenden Ausgangssignale auf die Feld-Schaltgeräte. Der Vorgang des Lesens von Eingangssignalen, die Logikbearbeitung auf der Basis der Eingangsdaten und das Aktualisieren der Ausgangsgeräte wird *Zyklus* genannt.

Flash-Speicher
In der CPU befindet sich auch eine Flash-Speicher-Komponente, in der sich das Betriebssystem der SPS befindet. Der Inhalt des Flash ist nicht-flüchtig; es ist daher keine Batteriepufferung erforderlich.

Das im Flash befindliche Betriebssystem ist eine Zusammenstellung der Kontrollprogramme, die der SPS ihre Identität geben, indem sie:

- ☐ die Sprache definieren, in der das Anwendungsprogramm geschrieben wird d.h. die Kontaktplanlogik
- ☐ den CPU-Speicher für bestimmte Zwecke verwalten
- ☐ die Struktur festlegen, in der die SPS Daten speichert und bearbeitet.

Das Kontaktplanlogik-Betriebssystem definiert die Funktionen der Modicon Micro-SPS. Dieser Funktionsumfang steht im Mittelpunkt dieses Buches.

Ausgänge
Die Ausgänge befinden sich im Anschlußblock an der Unterseite der SPS. Ausgänge schalten die vorhandenen Steuerspannungen, die das Feldgerät in Ihrer Anwendung betreiben oder anhalten. Wenn ein Ausgang durch die CPU auf EIN geschaltet wird, wird die Steuerspannung eingeschaltet und aktiviert das angesprochene Gerät.

Systemleistung

Zykluszeit
Die Zeit, die die CPU benötigt, um die Kontaktplanlogik des Programms zu lösen und die ihrer Kontrolle unterliegende Eingabe/Ausgabe zu aktualisieren, wird *Zykluszeit* genannt. Die Zykluszeit umfaßt die Logik-Bearbeitungszeit, die Zeit für die Bedienung der Ein-/Ausgänge und die Zeit, die für die Systemvorgänge benötigt werden.

Die für eine SPS maximal zulässige Zeit für einen einmaligen Zyklusdurchlauf des Logikprogramms beträgt 250 ms. Wenn der Zyklus in diesem Zeitraum nicht vollständig durchgelaufen ist, stoppt ein *Watchdog-Timer* in der CPU die Anwendung und sendet eine Zeitabschaltfehler-Meldung zum Programmiergerät. Durch diese Zykluszeit-Obergrenze wird verhindert, daß die SPS in unkontrollierte Zustände im Logikprogramm gerät.

Logik-Bearbeitungszeit
Die Zeit, die die CPU benötigt, um die Steuerlogik des Programms abzuarbeiten, abzüglich aller für Dienst- oder Serviceaufgaben benötigten Zeit, wird *Logik-Bearbeitungszeit* genannt.

- ☐ Die Logik-Bearbeitungszeit für die Micro SPS 110CPU311 und 110CPU411 beträgt 4.25 ms pro Tausend belegte Zellen in der Kontaktplanlogik
- ☐ Die Logik-Bearbeitungszeit für die Micro SPS 110CPU512 und 110CPU612 beträgt 2.5 ms pro Tausend belegte Zellen in der Kontaktplanlogik

Speicherverwaltung

Das Kontaktplanlogik-Betriebssystem legt fest, wie die zu Verfügung stehenden Speicherplätze einer Modicon Micro-SPS verwaltet werden. Es teilt den verfügbaren Systemspeicher in drei Speicherarten auf:

- *Anwender-Datenspeicher* — für Variablendaten, die sich während des Programmablaufs ändern
- *System-Konfigurationsspeicher* — für die Speicherung von Systemdaten-Tabellen wie E/A-Verzeichnis und SPS-Einstellungen
- *Anwender-Programmspeicher* — in dem das Kontaktplan-Logikprogramm erstellt und bearbeitet wird

Anwender-Datenspeicher

Die SPS stellt einen Bezug zwischen jedem Eingangs- und Ausgangssignal des Steuerungsvorgangs und einer Referenzzahl her, die in der Anwender-Datenspeichertabelle gespeichert ist und daher im Logikprogramm benutzt werden kann. (Diese Anwender-Datenspeichertabelle wird auch als *Status-RAM*-Tabelle bezeichnet.)

- Die SPS 110CPU311 und 110CPU411 haben 512 Worte Anwender-Datenspeicher
- Die SPS 110CPU512 und 110CPU612 haben 2048 Worte Anwender-Datenspeicher

Numerierung der Referenzen

Für die Handhabung der Ein- und Ausgabeinformation und der internen Logik benutzen die Modicon Micro-Steuerungen ein System von Referenznummern für die Logik-Programmierung. Jede Referenznummer hat als erstes eine Ziffer, die den E/A-Datentyp angibt, auf diese erste Ziffer folgt eine Gruppe von vier Ziffern, die den eindeutigen Ort dieses E/A-Punktes im Anwender-Datenspeicher angibt.

Es gibt vier Referenztypen:

Nummerierungssystem der E/A-Referenzen

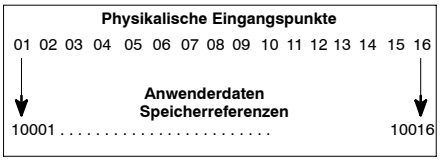
Referenz-zahl	Beschreibung
0*xxxx*	Ein diskreter Ausgang (oder eine Spule). Eine 0*x* Referenz kann dafür benutzt werden, reale Ausgangsdaten durch eine Ausgangseinheit des Steuerungssystems zu steuern oder eine oder mehrere Spulen im Status-RAM zu definieren. Eine bestimmte 0*x* Referenz kann nur einmal als Spule in einem Logikprogramm benutzt werden, aber dieser Spulenstatus kann beliebig oft für die Steuerung von Kontakten im Programm herangezogen werden.
1*xxxx*	Ein diskreter Eingang. Der EIN/AUS Status einer 1*x*-Referenz wird durch die Felddaten gesteuert, die an die CPU von einem Eingabegerät gesendet werden. Er kann für die Steuerung von Kontakten in einem Logikprogramm benutzt werden.
3*xxxx*	Ein Eingangregister. Ein 3*x* -Register enthält Informationen, die von einer externen Quelle, wie z.B. einem Drehschalter, einem Analogsignal oder Daten von einem Hochgeschwindigkeitszähler stammen und durch eine 16-Bit Zahl repräsentiert werden. Ein 3*x*-Register kann auch 16 aufeinanderfolgende, diskrete Eingangsignale beinhalten, die in das Register im Binär- oder BCD (binary coded decimal)-Format eingelesen wurden.
4*xxxx*	Ein Ausgans- oder Halteregister. Ein 4*x*-Register kann für die Speicherung numerischer Daten (binär oder dezimal) im Status-RAM benutzt werden, oder auch für das Senden der Daten von der CPU an eine Ausgangseinheit im Steuerungssystem.

Hinweis:
Das *x*, das nach der ersten Ziffer jedes Referenztyps steht, repräsentiert einen vierstelligen Speicherort im Anwender-Datenspeicher, z.B. gibt die Referenz 40201 an, daß die Referenz ein 16 Bit Ausgangs- oder Halteregister bezeichnet, an der Adresse 201 des Status-RAM.

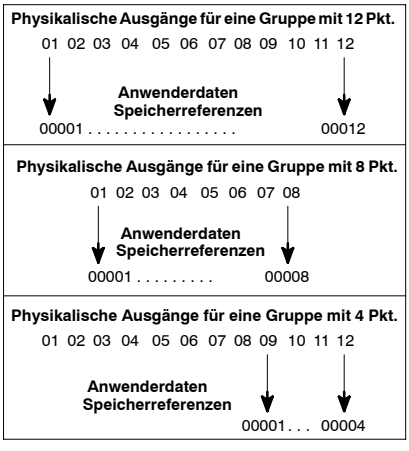
Jedes Wort des Anwender-Datenspeichers ist 16 Bit lang. Der (EIN/AUS) Zustand jedes diskreten E/A-Punktes wird durch den Wert 1 oder 0 repräsentiert, der einem einzelnen Bit in

einem Wort zugeordnet ist (16 0*x*- oder 1*x* -Referenzen pro Wort).

Für die E/A-Verzeichnisse wird der physikalische Eingabepunkt der internen Spule zugeordnet, die die niedrigste Nummer in der ersten 16'er Gruppe hat, der physikalische Eingabepunkt #2 der nächst höheren internen Spule, usw., wie nachstehend gezeigt:


Physikalische Eingangspunkte
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Anwenderdaten
Speicherreferenzen
10001 10016

Diskrete Ausgänge werden auf vergleichbare Weise entsprechend ihren Gruppierungen zugeordnet:


Physikalische Ausgänge für eine Gruppe mit 12 Pkt.
01 02 03 04 05 06 07 08 09 10 11 12
Anwenderdaten
Speicherreferenzen
00001 00012
Physikalische Ausgänge für eine Gruppe mit 8 Pkt.
01 02 03 04 05 06 07 08
Anwenderdaten
Speicherreferenzen
00001 00008
Physikalische Ausgänge für eine Gruppe mit 4 Pkt.
01 02 03 04 05 06 07 08 09 10 11 12
Anwenderdaten
Speicherreferenzen
00001. . . 00004

Bei analogen E/A wird jeder Eingangskanal und jeder Ausgangskanal einem ganzen Wort des Anwender-Datenspeichers zugeordnet (3*x*-Register für Eingänge und 4*x*-Register für Ausgänge).

System-Konfigurationsspeicher

Die im Systemspeicher enthaltene SPS-Konfiguration ist ein Schlüsselelement des Systems. Die Information der Konfiguration beinhaltet Angaben über

- ☐ die Betriebsart der SPS = Single, Parent oder Child
- ☐ die Parameter für die Kommunikations-Ports der SPS
- ☐ die Bereiche der für die Programmierung verfügbaren 0*x*, 1*x*, 3*x*, und 4*x*-Referenzen
- ☐ die Anzahl der von der SPS unterstützten E/A-Bereiche

Zugang zur Konfiguration haben Sie über die Software Ihres Programmiergerätes, mit der Sie viele dieser Parameter verändern können.

Der System-Konfigurationsspeicher ist so voreingestellt, daß er die folgende Standard-SPS-Konfiguration unterstützt:

SPS-Standard-Einstellungen

Parameter	Modell 110CPU	
	311 / 411	512 / 612
Anzahl der 0x Ausgänge	1024	1536
Anzahl der 1x Eingänge	256	512
Anzahl der 3x Eingänge	32	48
Anzahl der 4x Ausgänge	400	1872
Anzahl der E/A- Bereiche	5	5
Anzahl der Kontaktplanlogik-Segmente	2 (eines für die Standard-Kontaktplanlogik und eines für Interrupts und Unterprogramme)	2 (eines für die Standard-Kontaktplanlogik und eines für Interrupts und Unterprogramme)

Diese Standardwerte nutzen beim Setup den gesamten, verfügbaren SPS-Speicher aus. Teile der Systemkonfiguration können Sie ggf. neu definieren, um den E/A-Anforderungen einer speziellen Anwendung zu entsprechen.

Wenn Sie zum Beispiel eine 110CPU31101 SPS benutzen und Ihre

Anwendung anstelle der Standardkonfiguration von 32 Registern 35 Registereingänge benötigt, können Sie die zusätzlich benötigten drei Worte von einer anderen Stelle der Setup-Tabelle umwidmen. So könnten Sie, falls die Anwendung nicht alle 1024 Ausgänge benötigt, in der Setup-Tabelle 976 diskrete Ausgänge festlegen und die dadurch frei werdenden 48 Bit als drei zusätzliche (16-Bit) Eingangs-Registerworte nutzen.

☞ **Hinweis** Der im SPS-Setup konfigurierte Gesamtspeicherplatz kann nicht die Summe der Werte übersteigen, die in der Tabelle für die standardmäßigen Setup-Werte der SPS angezeigt wird.

Anwender-Programmspeicher

Abhängig von dem von Ihnen benutzten SPS-Modell beträgt der verfügbare Speicher für die Kontaktplanlogik:

- □ 1024 Worte (bei den SPS 110CPU311 und 110CPU411)
- □ 2048 Worte (bei den SPS 110CPU512 und 110CPU612)

Dabei handelt es sich um den Gesamtspeicher, der für die Programmlogik verfügbar ist. Einige optionale SPS-Funktionen, z.B. zusätzliche, ladbare Befehle, nehmen allerdings bereits einen Teil des für die Programmierung reservierten Anwenderspeichers in Anspruch.

Der Anwender-Programmspeicher ist in zwei *Segmente* aufgeteilt. In dem ersten Segment befindet sich die gesamte Kontaktplanlogik für die Standard Anwendungssteuerung. Das zweite ist für Logik-Unterprogramme reserviert, die entweder von einem Befehl mit dem Namen **JSR** der Kontaktplanlogik aufgerufen werden können, oder durch den Eingang eines Hochgeschwindigkeits-Interrupts (verfügbar bei den SPS 110CPU411, 110CPU512 und 110CPU612).

☞ **Hinweis** Mehr Information über Unterprogramme erhalten Sie im Kapitel 10.10

Es folgen die ladbaren Befehle, die bei den Modicon Micro-Steuerungen gebraucht werden können, mit der Angabe ihres Speicherbedarfs:

Loadable Name	Größe (Worte)	Standard Opcode	Funktion
EARS	760	5F	Zur Entwicklung eines Frühalarm-systems
EUCA	160	1F	Algoritmus zur Konvertierung technischer Ein-heiten
FN*xx*	Anwen-der-definiert	5F	Zur kundenspezi-fischen Entwick-lung eines DX-Loadables

Mehr Informationen über diese ladbaren Befehle können Sie den folgenden technischen Publikationen von Modicon entnehmen:

- □ ***Frühalarmsystem Benutzerhandbuch*** (GM-EARS-001)
- □ ***EUCA ladbarer Funktionsblock Benutzerhandbuch*** (GM-EUCA-001)
- □ ***Kundenspezifische DX-Loadables Software Programmierhandbuch*** (GM-CLSS-001)

Speicher-Pufferung

Anwender-Datenspeicher, Anwender-Programmspeicher und System-Konfigurationsspeicher können auf eine von drei verschiedenen Weisen gesichert werden:

- ☐ Mit einer optionalen Lithiumbatterie (110XCP98000)
- ☐ Mit einem optionalen Batteriekondensator (110XCP99000)
- ☐ In einem reservierten Bereich des Flash-Speichers

Optionale Pufferung-Verfahren

Beim Gebrauch einer Lithiumbatterie oder eines Batteriekondensators wird der aktuelle Speicherinhalt automatisch im Falle eine Stromausfalls gesichert. Wird die Stromversorgung wiederhergestellt, kann die SPS ihren Betrieb mit der Konfiguration und den Programmwerten wieder aufnehmen, die vor Eintritt des Stromausfalls vorhanden waren.

Die Batterie kann die Daten für ein Jahr, der Batteriekondensator ein typisches Benutzerlogikprogramm für 72 Stunden sichern (mehr Details finden Sie im Installationshandbuch Ihrer SPS).

Pufferung mit Hilfe des Flash-Speichers

Ein Teil des Flash-Speichers aller Modicon Micro-SPS ist für die Speicherung der Systemkonfiguration, der Anwenderlogik und des Anwender-Datenspeichers reserviert. Dadurch können Sie Ihre Konfiguration und Anwenderlogik sogar auch dann sichern, wenn Sie keine Batterie oder keinen Batteriekondensator zur Pufferung einsetzen.

Um den Speicher im Flash zu sichern, müssen Sie den Befehl *PROM/Flash speichern* von der Software Ihres Programmiergeräts geben *(Pfad Online → SPS-Funktionen).* Es werden nur die Werte im Flash gesichert, die im Speicher beim Geben des Befehls vorliegen. Eine *Speichern auf Flash*-Operation ist nur bei einer SPS zulässig, die konfiguriert wurde und gestoppt ist, d.h. nicht während des Abarbeitens des Logikprogramms.

Wenn nach einem Stromausfall der Speicherinhalt vom Flash wieder in der SPS hergestellt wird, werden die Werte wiedereingesetzt, die beim letzten Speichern-Befehl aktuell waren.

SPS Einschalt-Prüfung

Nachdem eine SPS eingeschaltet wurde, prüft sie zuerst den System-Konfigurationsspeicher, um festzustellen, ob eine gültige Konfiguration vorhanden ist. Wenn eine gültige Konfiguration durch die optionale Batteriepufferung gespeichert wurde, werden diese Daten in den Anwender-Datenspeicher übernommen. Die SPS wird sich selbst mit diesen Werten konfigurieren und dann betriebsbereit sein.

Findet die SPS im Anwender-Datenspeicher keine gültige Konfiguration, wird sie den Flash-Pufferungbereich prüfen. Wenn eine gültige Konfiguration im Flash gesichert wurde, wird sich die SPS selbst mit diesen Werten konfigurieren und dann betriebsbereit sein.

Findet die SPS weder im Speicher noch im Flash eine gültige Konfiguration, wird sie in einem *unkonfigurierten* Zustand starten. Bevor die SPS programmiert werden kann oder Logik bearbeiten kann, müssen Sie in diesem Fall ein Programmiergerät an der SPS anschließen und sie konfigurieren.

Lagerung einer SPS mit im Flash gespeicherter Anwenderlogik

☞ **Hinweis** Wenn Sie ein Logikprogramm im Flash gespeichert haben und die SPS außer Dienst stellen, müssen Sie bedenken, daß alle im Flash gespeicherten Werte nicht-flüchtig sind.

Wenn die SPS irgendwann in der Zukunft wieder eingeschaltet wird, startet sie sofort mit dem gespeicherten Programm. Es besteht daher eine potentielle Gefahrenquelle, falls die SPS nach einer Langzeitlagerung in einer neuen Installation eingesetzt wird.

Wenn Sie nicht sicher sind, wie die SPS zukünfig eingesetzt wird, könnte es sinnvoll sein, vor der Entfernung die Logik aus dem Flash der SPS zu löschen. Gehen Sie dafür wie folgt vor:

Schritt 1. Löschen Sie alle Netzwerke des Logikprogramms.

Schritt 2. Setzen Sie alle Konfigurationsparameter der SPS auf ihre Standardwerte.

Schritt 3. Stellen Sie sicher, daß die SPS gestoppt ist.

Schritt 4. Geben Sie dann mit Ihrer Programmiergerät--Software den Befehl *Speichern auf Flash*.

SPS Betriebsarten

Eine Modicon Micro-SPS kann für eine der drei folgenden Betriebsarten konfiguriert werden:

- □ *Einzel*-Modus — Die SPS arbeitet als eigenständiges, programmierbares Steuerungssystem, das seine eigenen, fest installierten E/A-Resourcen verwaltet (im Fall der SPS 110CPU512 und 110CPU612 kann die SPS zusätzliche A120 E/A-Resourcen verwalten).
- □ *Parent*-Modus — Die SPS arbeitet als die einzige SPS in einer E/A-Erweiterungskopplung, deren CPU die fest eingebauten E/A-Ressourcen aller SPS in dieser Erweiterungskopplung verwaltet.
- □ *Child*-Modus — Die SPS arbeitet als eine SPS in einer E/A-Erweiterungskopplung, und unterstellt einen Teil oder ihre gesamten eingebauten E/A-Ressourcen dem Zugriff und der Verwaltung der Parent-SPS dieser Erweiterungskopplung.

Die E/A-Erweiterungskopplung

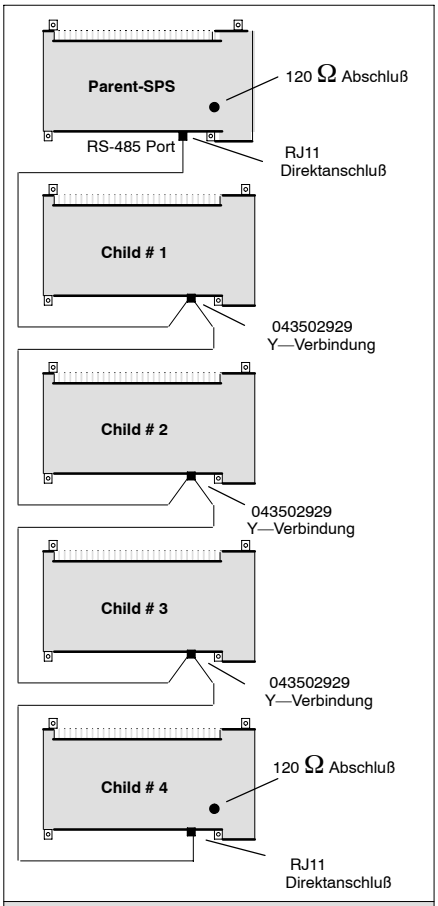
Eine *E/A-Erweiterungskopplung* besteht aus einer Parent-SPS und 1 ... 4 Child-Steuerungen, die über normales, sechspoliges Telefonkabel verbunden sind. Jedes Kabel hat eine RJ11 Steckverbindung an beiden Enden. SPS-zu-SPS Verbindungen werden über den RS-485 (**exp link**)-Port jeder Einheit hergestellt.

Nur eine SPS innerhalb der Verbindung kann als Parent konfiguriert sein. Alle anderen SPS in der Verbindung müssen als Child-SPS konfiguriert sein. Eine SPS mit der Betriebsart Single kann nicht in der Erweiterungskopplung betrieben werden.

Jede Child SPS wird mit einer Child ID # im Bereich von #1 ... #4 eindeutig adressiert. Auf die festen E/A-Ressourcen der Child-SPS kann von der Logik der Parent zugegriffen und diese auch gesteuert werden.

☞**Hinweis** Sie als Anwender sind dafür verantwortlich, sicherzustellen, daß jede Child-SPS eine eindeutige Child ID-Nummer erhält. Die Zuweisung dieser Child-Kennummer geschieht durch Anschluß des Programmiergeräts und Eingabe der Nummer als Teil der Konfiguration des Child.

Die E/A-Erweiterung geschieht über serielle, Punkt-zu-Punkt Verbindungen zwischen der Parent-SPS und den Child-SPS, wie unten dargestellt.



Ein E/A-Erweiterungsnetz

A120 E/A-Erweiterung

Die SPS 110CPU512 und 110CPU612 sind mit einem 30-poligen Erweiterungsport ausgestattet, der es den Einheiten ermöglicht, mit Baugruppenträgern der A120 Eingabe/Ausgabe zu kommunizieren. Dieser Port dient speziell der A 120 E/A-Kommunikation.

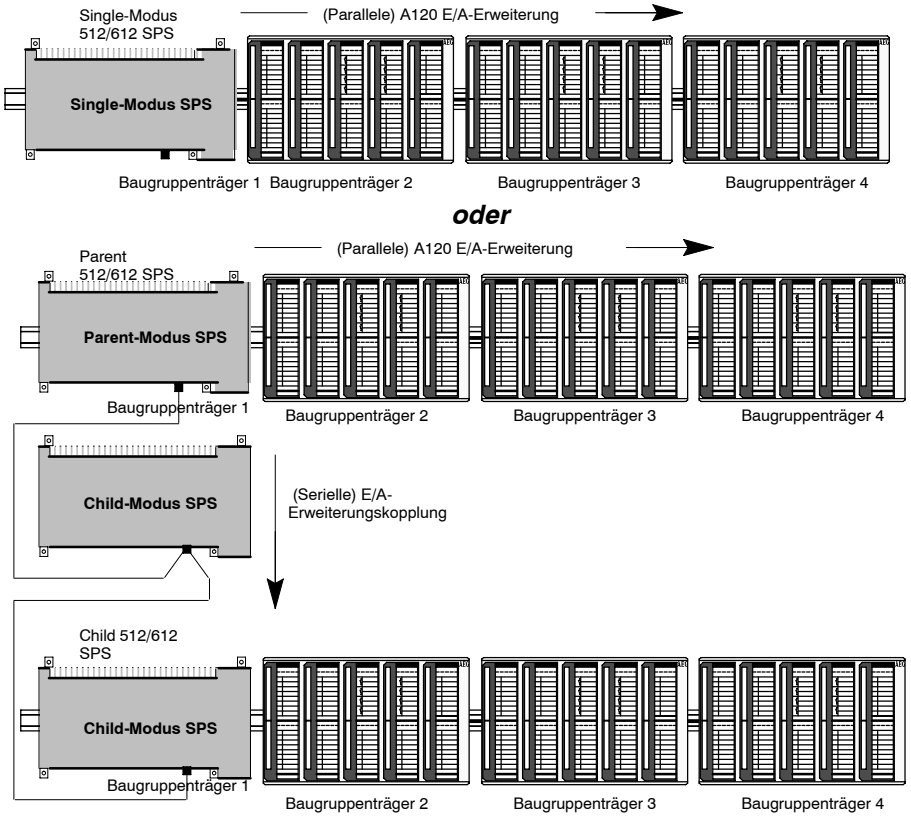
☞ **Hinweis** Die SPS 110CPU311 und 110CPU411 unterstützen nicht die A120 E/A-Erweiterung.

Bei der A120 E/A-Erweiterung werden 2 ... 4 Baugruppenträger über einen parallelen Bus, der auf einer DIN-Schiene montiert wird, miteinander verbunden. Die Steuerung selbst wird immer als Baugruppenträger 1 konfiguriert, die A120 E/A-Gehäuse als Baugruppenträger 2...4.

Eine A120 E/A-Erweiterung kann von der Steuerung (single, parent, child) mit jeder der drei Betriebsarten genutzt werden.

Ein Zugriff auf die A120 E/A kann nur durch die SPS erfolgen, an die die E/A angeschlossen ist. Das bedeutet, daß das Logikprogramm, welches die A120 E/A und alle mit ihr assoziierten E/A-Verzeichnisse steuert, in der SPS gespeichert sein muß, an die die A120 E/A gekoppelt ist.

☞ **Hinweis** Wenn eine Child-SPS in einer seriellen E/A-Erweiterungskopplung eine A120 E/A-Erweiterung nutzt, kann die Parent in dieser Kopplung nicht auf die A120 E/A-Erweiterung dieses Child zugreifen. Die Child muß mit ihrer eigenen Kontaktplanlogik, SPS-Konfiguration und dem E/A-Verzeichnis unabhängig programmiert sein.


Single-Modus
512/612 SPS
(Parallele) A120 E/A-Erweiterung
Single-Modus SPS
Baugruppenträger 1
Baugruppenträger 2
Baugruppenträger 3
Baugruppenträger 4
oder
Parent
512/612 SPS
(Parallele) A120 E/A-Erweiterung
Parent-Modus SPS
Baugruppenträger 1
Baugruppenträger 2
Baugruppenträger 3
Baugruppenträger 4
Child-Modus SPS
(Serielle) E/A-
Erweiterungskopplung
Child 512/612
SPS
Child-Modus SPS
Baugruppenträger 1
Baugruppenträger 2
Baugruppenträger 3
Baugruppenträger 4

Kontaktplanlogik-Befehlsvorrat

Das Kontaktplanlogik-Betriebssystem befindet sich im Flash-RAM der Modicon Micro SPS. Beachten Sie, daß einige Micro-Modelle einen erweiterten Befehlssatz haben, mit Funktionen, über die die einfacheren Modelle nicht verfügen.

Standard Kontaktplanlogik-Befehlsvorrat (verfügbar in allen Micro-SPS)	
Befehl	**Beschreibung**
Schaltlogik	
	Schließer
	Öffner
	Positiver Wischer
	Negativer Wischer
	Normale Spule
	Speicherresidente Spule
Zähler	
UCTR	Vorwärtszähler von 0 bis zu einem spezifizierten Wert
DCTR	Abwärtszähler von 0 bis zu einem spezifizierten Wert
Timer	
T1.0	Timer mit einer Sekunden-Inkrementierung
T0.1	Timer mit einer Zehntelsekunden-Inkrementierung
T.01	Timer mit einer Hunderstelsekunden-Inkrementierung
T1MS	Timer mit einer ms-Inkrementierung
Integer Math	
ADD	Addition
SUB	Subtraktion oder größer/kleiner-Operationen
MUL	Multiplikation
DIV	Division
Datenverschiebung	
R→T	Register-zu-Tabelle Verschiebung
T→R	Tabelle-zu-Register Verschiebung
T→T	Tabelle-zu-Tabelle Verschiebung
BLKM	Blockverschiebung
FIN	First-In Operation in eine Warteschlange
FOUT	First-Out Operation aus einer Warteschlange
SRCH	Suche in einer Tabelle nach einem Bitmuster in einem der Register

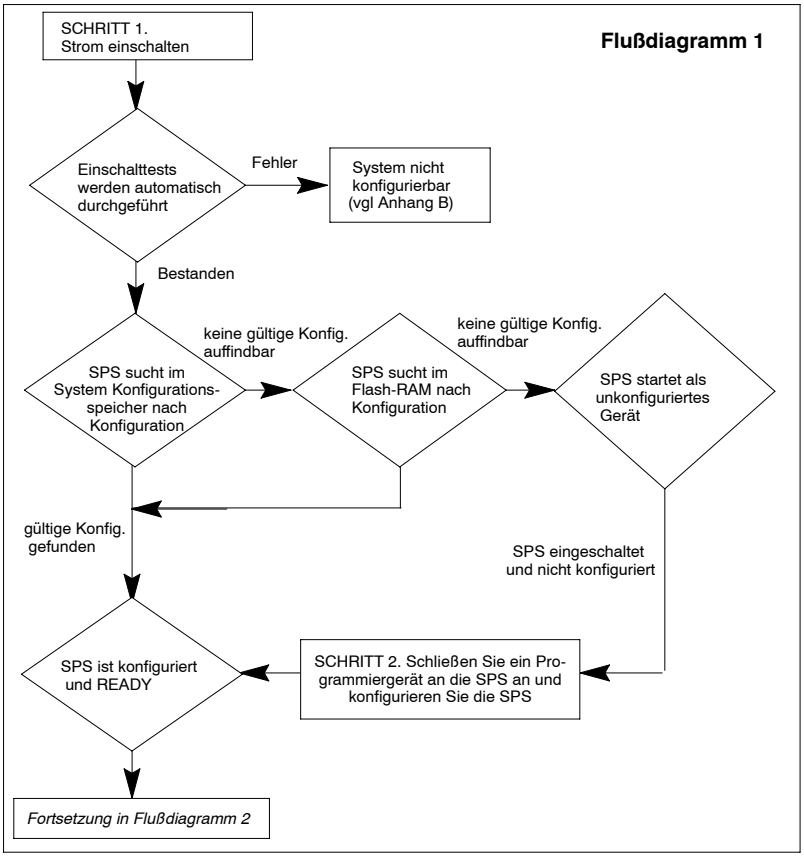
Standard Kontaktplanlogik- Befehlsvorrat (Fortsetzung)	
Befehl	**Beschreibung**
Datenmatrix	
UND	Logisches UND zweier Matrizen
ODER	Logisches ODER zweier Matrizen
XODER	Logisches exklusives ODER zweier Matrizen
COMP	Logisches Komplement des Bitmusters einer Matrix
CMPR	Logischer Vergleich der Bitmuster zweier Matrizen
MBIT	Bit-Änderung, d.h. Änderung des aktuellen Wertes (1,0) eines Bits
SENS	Bit-Inhalt, d.h. Anzeige des aktuellen Inhalts (1,0) eines Bits
BROT	Bit-Rotation, d.h. Verschiebung der Bitpositionen nach links oder rechts in der Matrix
ASCII	
COMM	ASCII Lese- oder Schreib-Kommunikationsvorgang
Ordnen	
SCIF	Drum-Sequence- und Eingabevergleichs-Operationen
Unterprogramme	
JSR	Springt aus dem Logikzyklus der Steuerlogik zu dem Kontaktplanlogik-Unterprogramm, das im letzten Segment programmiert wurde
LAB	Bezeichnet die Einstiegsposition des im letzten Segment aufgerufenen Unterprogramms
RET	Führt den Logikzyklus wieder an die vorherige Stelle in der Logik vor Beginn der Ausführung des JSR
CTIF	Einrichtung der Hochgeschwindigkeits-Eingänge für Interrupt/Zähler/ Timer-Operationen
Andere	
STAT	Prüft und stellt dar den Zustand der SPS und ihrer E/A
SKP	Veranlaßt den Logikzyklus, bestimmte Netzwerke im Programm zu überspringen

Erweiterte Kontaktplanlogik-Befehle (nur verfügbar in bestimmten Modellen, 110CPU512 und 110CPU612)	
Befehl	**Beschreibung**
BLKT	Block-zu-Tabelle Verschiebung
TBLK	Tabelle-zu-Block Verschiebung
CKSM	Führt CRC-16, LRC, lineare oder binäre Checksum-Prüfung durch
PID2	Führt proportionale, integrale, differentiale, Steuerungsfunktionen aus
EMTH	Führt erweiterte, mathematische Funktionen aus, wie Quadratwurzel, Prozeß-Quadratwurzel, Logarithmus, Antilogarithmus und Fließpunktoperationen aus

Kapitel 2
Start-Prozeduren

- ☐Inbetriebnahme
- ☐Parameter der automatischen Konfiguration
- ☐Automatische Konfigurierung der Kommunikations-Ports
- ☐Modifizierung der Konfigurationsparameter
- ☐Adressierung der E/A-Bereiche
- ☐Adressierung der A120 E/A
- ☐Adressierung der E/A in einer Erweiterungskopplung
- ☐Aufteilung lokaler E/A zwischen Parent- und Child-SPS
- ☐Generalisierter Datentransfer
- ☐SPS-Betriebsoperationen

Inbetriebnahme


SCHRITT 1.
Strom einschalten
Flußdiagramm 1
Einschalttests werden automatisch durchgeführt
Fehler
System nicht konfigurierbar (vgl Anhang B)
Bestanden
SPS sucht im System Konfigurations-speicher nach Konfiguration
keine gültige Konfig. auffindbar
SPS sucht im Flash-RAM nach Konfiguration
keine gültige Konfig. auffindbar
SPS startet als unkonfiguriertes Gerät
gültige Konfig. gefunden
SPS eingeschaltet und nicht konfiguriert
SPS ist konfiguriert und READY
SCHRITT 2. Schließen Sie ein Pro-grammiergerät an die SPS an und konfigurieren Sie die SPS
Fortsetzung in Flußdiagramm 2

Einschalten der Stromversorgung

Sobald bei einer Modicon Micro SPS die Stromversorgung hergestellt wird, versucht sie den Betrieb aufzunehmen. Das Betriebssystem sucht im Speicher nach dort vorher gespeicherten Konfigurationsdaten.

Starten einer bereits konfigurierten SPS

Wenn die SPS bereits früher in Betrieb war und eine gültige Konfiguration (eventuell auch ein Logikprogramm) in ihrem Speicher hat, wird die SPS sofort den Betrieb mit diesen gespeicherten Werten aufnehmen.

Hat die SPS ein optionales, batteriegepuffertes Backup, werden die bestehenden Konfigurationsparameter in ihrem System-Konfigurationsspeicher und die bestehenden Logikwerte im Anwender-Programmspeicher gefunden. Alternativ dazu können die Konfiguration und die Anwenderlogik auch im Flash-RAM der SPS gespeichert werden, wenn Sie kein Batteriebackup durchführen.

Wie das obige Flußdiagramm zeigt, prüft das Betriebssystem der SPS zuerst den Konfigurationsspeicher. Wenn dort eine gespeicherte, gültige Konfiguration

gefunden wird, werden diese Werte für den Betrieb benutzt. Wird keine gültige Konfiguration gefunden, wird im Flash-RAM der SPS nach einer gültigen Konfiguration gesucht. Wenn dort eine gespeicherte, gültige Konfiguration gefunden wird, werden diese Werte für den Betrieb benutzt.

Wenn die SPS sich vorher im RUN-Modus befunden hatte, nimmt die SPS unverzüglich die Logikbearbeitung auf. Ein Programmiergerät braucht dann nicht angeschlossen zu werden.

Befand sich die SPS vorher im Modus STOP, müssen Sie ein Programmiergerät an einen der **Comm** Ports der SPS anschließen, um diese zu starten.

Das Starten einer unkonfigurierten SPS

Wenn ein Betriebssystem keine gültige Konfiguration im Flash der SPS oder im System-Konfigurationsspeicher der SPS finden kann, startet sie als *unkonfiguriertes* Gerät. Beim ersten Einschalten überhaupt startet eine SPS unkonfiguriert, ebenso, wenn die Konfigurationswerte gelöscht wurden oder defekt sind.

Sie müssen die SPS konfigurieren, bevor Sie ein Logikprogramm schreiben oder E/A-Vorgänge ausführen können.

Konfigurierung einer Modicon Micro SPS

Schritt 1. Schließen Sie ein Programmiergerät, wie z.B. einen PC mit MODSOFT Lite oder das HHP an einen RS-232 Port der SPS an.

Schritt 2. Gehen Sie im Menüsystem des Programmiergeräts zum Konfigurationseditor. (Der Pfad zum Konfigurationseditor hängt vom benutzten Gerät ab, es ist jedoch überall ein Bildschirm oberster Ebene, der über wenige Tastenbetätigungen aufgerufen werden kann.)

Schritt 3. Vergewissern Sie sich, daß das Programmiergerät für die Konfigurierung des richtigen SPS Modells eingestellt ist (z.B eines 110CPU31101 oder eines110CPU51200). Das HHP (= Handheld-Panel) zeigt diese Information automatisch beim Einschalten an. Bei MODSOFT Lite werden Sie gebeten, das Modell aus einer Auswahlliste zu wählen.

Schritt 4. Wählen Sie die gewünschte Betriebsart der zu konfigurierenden SPS. Die Betriebsart kann entweder Single, Parent oder Child sein.

Schritt 5. Übertragen Sie die Konfigurationsparameter vom Programmiergerät zur SPS.

Ergebnis. Das Programmiergerät konfiguriert die SPS automatisch mit einem kompletten Satz gültiger Parameter auf der Basis des von Ihnen angegebenen Modelltyps und der Betriebsart. Jetzt ist die SPS konfiguriert.

Parameter der automatischen Konfiguration

Auf der Basis des von Ihnen angegebenen Modelltyps und der Betriebsart konfiguriert das Programmiergerät automatisch die SPS mit einem kompletten Satz gültiger Parameter. Die *Autokonfigurations-Parameter* werden in den folgenden drei Tabellen gezeigt.

Autokonfiguration einer SPS in der Betriebsart Single

Wenn Sie eine SPS in der Betriebsart Single konfigurieren, benötigen Sie lediglich die unten angeführten automatisch konfigurierten Parameter, um mit der Kontaktplanlogik-Programmierung zu beginnen.

Autokonfigurations-Parameter für eine SPS im Single-Modus

Parameter	110CPU Modelle	
	311 / 411	**512 / 612**
Anzahl der 0*x* Referenzen	1024	1536
Anzahl der 1*x* Referenzen	256	512
Anzahl der 3*x* Referenzen	32	48
Anzahl der 4*x* Referenzen	400	1872
Anzahl der Kontakt-planlogik-Segmente	2 (das erste für Steuerlogik, das zweite für Unterprogramme)	2 (das erste für Steuerlogik, das zweite für Unterprogramme)
RS-232 Port (**comm 1**)	Dezierter Modbus-Modus: 8-Bit RTU Kommunikation, 9600 Baud, gerade Parität, 1 Stopbit, Modbus-Adresse #1	Dedizierter Modbus-Modus: 8-Bit RTU Kommunikation, 9600 Baud, gerade Parität, 1 Stopbit, Modbus-Adresse #1
RS-232 Port (**comm 2**)	nicht verfügbar	Dedizierter Modbus-Modus: 8-Bit RTU Kommunikation, 9600 Baud, gerade Parität, 1 Stopbit, Modbus-Adresse #1
RS-485 Port (**exp. link**)	Dedizierte ASCII 8-Bit ASCII Kommunication, 9600 baud, gerade Parität, 1 Stopbit	Dedizierte ASCII 8-Bit ASCII Kommunikation, 9600 Baud, gerade Parität, 1 Stopbit

Autokonfiguration einer SPS in der Betriebsart Parent

Wenn Sie die Betriebsart Parent wählen, muß die Anzahl der Child-SPS angegeben werden, die in der E/A-Erweiterungskopplung zugelassen sind. Die Zahl muß im Bereich von 1 bis 4 liegen.

Sobald Sie diese Zahl eingegeben haben, kann die SPS programmiert werden.

Autokonfigurations-Parameter einer SPS in der Betriebsart Parent

Parameter	110CPU Modelle	
	311 / 411	**512 / 612**
Anzahl der 0*x* Referenzen	1024	1536
Anzahl der 1*x* Referenzen	256	512
Anzahl der 3*x* Referenzen	32	48
Anzahl der 4*x* Referenzen	400	1872
Anzahl der Parent-Steuerungen in der E/A-Erweiterungskopplung	*muß vom Anwender definiert werden*	*muß vom Anwender definiert werden*
Anzahl der Kontaktplanlogik-Segmente	2 (das erste für Steuerlogik, das zweite für Unterprogramme)	2 (das erste für Steuerlogik, das zweite für Unterprogramme)
RS-232 Port (**comm 1**)	Modbus/ASCII Toggle-Modus: 8-Bit RTU/8-Bit ASCII Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1	Dedizierter Modbus-Modus: 8-Bit RTU Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1
RS-232 Port (**comm 2**)	N/V	Modbus/ASCII Toggle-Modus: 8-Bit RTU/8-Bit ASCII Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1
RS-485 Port (**exp. net**)	E/A-Erweiterungsnetz: 9-Bit Datenkommunikation, 125 ,000 Baud, 1 Stopbit	E/A-Erweiterungsnetz: 9-Bit Datenkommunikation, 125 ,000 Baud, 1 Stopbit

Autokonfiguration einer SPS in der Betriebsart Child

Wenn Sie die Betriebsart Child wählen, muß der SPS eine Child Kenn-Nummer (ID) zugewiesen werden. Die Zahl muß im Bereich von 1 bis 4 liegen und sie muß einmalig sein, nur für dieses Child, das Sie konfigurieren, und darf bei keiner anderen Child SPS in dieser E/A-Erweiterungskopplung verwendet werden.

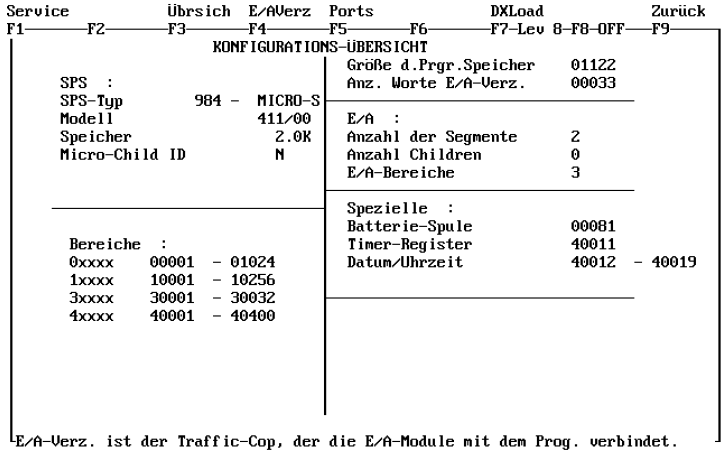
Sobald Sie diese Zahl eingegeben haben, kann die SPS programmiert werden.

Autokonfigurations-Parameter einer SPS in der Betriebsart Child

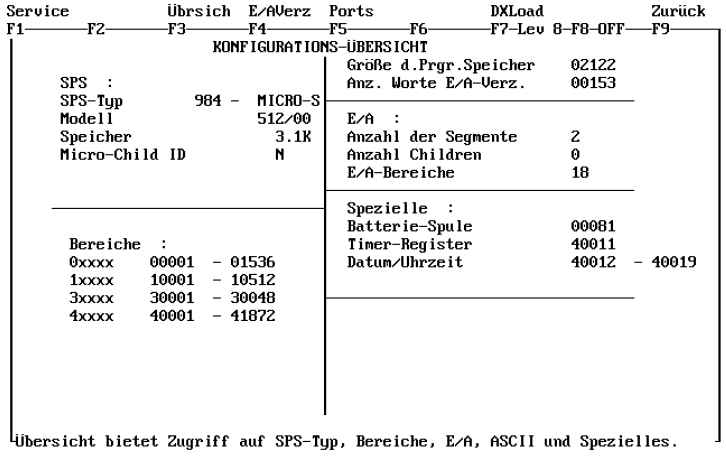
Parameter	110CPU Modelle 311 / 411	110CPU Modelle 512 / 612
Anzahl der 0*x* Referenzen	1024	1536
Anzahl der 1*x* Referenzen	256	512
Anzahl der 3*x* Referenzen	32	48
Anzahl der 4*x* Referenzen	400	1872
Child ID #	*muß vom Anwender definiert werden*	*muß vom Anwender definiert werden*
Anzahl der Kontakt-planlogik-Segmente	2 (das erste für Steuerlogik, das zweite für Unterprogramme)	2 (das erste für Steuerlogik, das zweite für Unterprogramme)
RS-232 Port (**comm 1**)	Modbus/ASCII Toggle-Modus: 8-Bit RTU/8-Bit ASCII Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1	Dedizierter Modbus-Modus: 8-Bit RTU Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1
RS-232 Port (**comm 2**)	N/V	Modbus/ASCII Toggle-Modus: 8-Bit RTU/8-Bit ASCII Kommunikation, 9600 Baud, gleiche Parität, 1 Stopbit, Modbus-Adresse #1
RS-485 Port (**exp. net**)	E/A-Erweiterungsnetz: 9-Bit Datenkommunikation, 125 ,000 Baud, 1 Stopbit	E/A-Erweiterungsnetz: 9-bit Datenkommunikation, 125 ,000 Baud, 1 Stopbit

Einige Beispiele für Autokonfigurationen

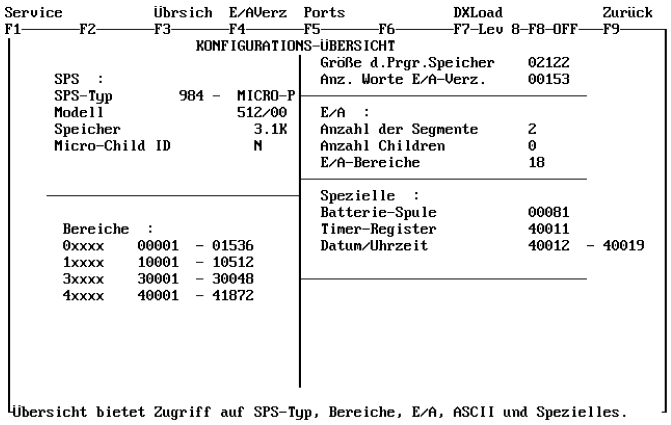
Im folgenden werden einige MODSOFT Lite Konfigurations-Übersichtsbildschirme dargestellt, wobei die Bedeutung der angezeigten Parameter erläutert wird. Unten werden drei Bildschirme für eine 110CPU51200 SPS gezeigt, mit jeder der drei verschiedenen Betriebsarten. Die MODSOFT Lite Beispiele dienen hier dazu, die der SPS Konfigurierung zugrundeliegenden Konzepte zu veranschaulichen. MODSOFT Lite ist nicht die einzige Programmiersoftware für die Konfigurierung einer Micro. Sie wird hier als Beispiel gebraucht, da die einzelnen Bildschirme mehr Werte enthalten als die des HHP. Eine ausführliche Beschreibung von MODSOFT Lite oder der Editierung mit dem HHP finden Sie im Programmierhandbuch Ihres Softwarepaketes.



Bildschirm 1. 110CPU51200 SPS mit Autokonfigurations-Parametern im Modus Single



Bildschirm 2. 110CPU51200 PLC SPS mit Autokonfigurations-Parametern im Modus Parent



Bildschirm 3. 110CPU51200 SPS mit Autokonfigurations-Parametern im Modus Child

SPS Betriebsart

Die Betriebsart wird im Eintrag `SPS-Typ` im oberen linken Datenfeld der Bildschirme angezeigt. `MICRO-S` steht für Single-Modus, `MICRO-P` für Parent-Modus und `MICRO-C` für Child-Modus.

Child-ID

Für eine SPS, die für die Betriebsart Child konfiguriert wurde, muß die Child Kenn-Nummer angegeben werden. Die MODSOFT Lite Konfiguration setzt standardmäßig die Nummer 1 ein. Falls Sie in einer Erweiterungskopplung mehr als einem Child konfigurieren, müssen Sie sich vergewissern, daß jedes eine eindeutige Kenn-Nummer im Bereich zwischen 1 und 4 hat.

Dieser Parameter findet bei Steuerungen mit der Betriebsart Single odem Parent keine Anwendung. Bei SPS, die in einer dieser beiden Betriebsarten arbeiten, wird bei `Micro Child ID` angegeben: `KEINE`.

0*x*, 1*x*, 3*x* und 4*x* Referenzbereiche

Der Bereich der internen Speicherreferenzen ist in allen Betriebsarten gleich. Die Bereichszuweisungen durch die Autokonfiguration stellen die maximal verfügbare Anzahl an Referenzen für Modell 110CPU51200 dar.

Hinweis Bei den Modellen 110CPU311 und 110CPU411 ist der Referenzbereich kleiner.

Anzahl der Kontaktplanlogik-Segmente

Durch die Autokonfiguration wird die Anzahl der Kontaktplanlogik-Segmente auf 2 eingestellt. Das erste Segmente steht für die normale Steuerungslogik zur Verfügung, das zweite Segment für die Logik von Unterprogrammen.

Anzahl des Child-Steuerungen
Wenn die SPS in der Betriebsart Parent konfiguriert ist, muß die Anzahl des Child-Steuerungen angegeben werden, auf die die Parent-Steuerung in dieser Erweiterungskopplung zugreifen kann. Die Voreinstellung von MODSOFT Lite ist 1. Verändern Sie diesen Parameter, wenn Sie mehr als ein Child in dieser Erweiterungskopplung anschließen.

Dieser Parameter findet bei SPS mit der Betriebsart Single oder Child keine Anwendung. Bei SPS, die in einer dieser beiden Betriebsarten arbeiten, wird bei `Anzahl der Children` angegeben:`0`.

E/A-Bereiche
Ein E/A-Bereich ist eine E/A-Einheit, die mit einem bestimmten Micro SPS-Typ verbunden ist. Diese E/A-Bereiche, die detailliert weiter unten in diesem Kapitel beschrieben werden,umfassen fest in die SPS eingebaute E/A und beliebige A 120 E/A-Module, die über den parallelen Port an die SPS angeschlossen sind.

☞ **Hinweis** Nur die Modelle110CPU512 und 110CPU612 unterstützen A120 E/A, die Modelle 110CPU311 und 110CPU411 jedoch nicht.

Die Modelle 110CPU512 verfügen alle standardmäßig über 18 E/A-Bereiche. Die Zahl ermöglicht Ihnen, drei oder vier lokale E/A-Bereiche, diskrete E/A, Hochgeschwindigkeitseingänge und die generalisierten Datentransfermöglichkeiten (werden später genauer erläutert) zu unterstützen, sowie bis zu 15 Steckplätze mit A120 E/A.

☞ **Hinweis** Die Modelle 110CPU311 und 110CPU411 haben standardmäßig eine sehr viel kleinere Anzahl von E/A-Bereichen, da diese Einheiten keine A120 E/A unterstützen.

Die Batteriespule
Das Betriebssystem reserviert automatisch Referenz 00081 als Batteriespule. Diese Spule arbeitet ganz ähnlich, wie die LED **batt low** an der SPS, denn sie schaltet sich EIN, wenn die Batterie ersetzt werden muß. Diese Spule kann mit einer externen Alarmierung oder Anzeige gekoppelt werden, die Sie auf das Ersetzen der Batterie aufmerksam machen.

Wenn die Batteriespule auf EIN schaltet, sollte die Batterie innerhalb von 14 Tagen ersetzt werden.

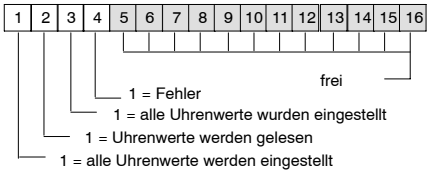
Das Timer-Register
Das Betriebssystem reserviert automatisch Referenz 40011 als freilaufenden Timer. Dieses Register steht Ihnen in der Kontaktplanlogik für 10 ms-Anwendungen zur Verfügung.

Die Uhr

Das Betriebssystem reserviert automatisch einen Block von acht aufeinanderfolgenden Ausgangsregistern (40012 ... 40019) für die Speicherung von Daten der SPS-Uhr (bei den Modellen 110CPU411, 110CPU512 und 110CPU612). Um die Uhr einsetzen zu können, muß sie zunächst initialisiert werden.

Die 16 Bit in jedem Register werden für die Speicherung der folgenden Informationen benutzt:

☐ Register 40012=Steuerregister:



☐ Register 40013=Wochentag (Sonntag = 1, Montag = 2, etc.)

☐ Register 40014=Monat (Jan. = 1, Dez. = 12)

☐ Register 40015=Tag im Monat (1 ... 31)

☐ Register 40016=Jahr (0 ... 99)

☐ Register 40017=Stunde (0 ... 23)

☐ Register 40018=Minute (0 ... 59)

☐ Register 40019=Sekunde (0 ... 59)

Wenn Sie die Uhr z.B. um 9:25:30 am Donnerstag, den 18 März 1993 auslesen, wird der Registerblock die folgende Information anzeigen:

40012 = 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
40013 = 5 (Dezimalformat)
40014 = 3 (Dezimalformat)
40015 = 18 (Dezimalformat)
40016 = 93 (Dezimalformat)
40017 = 9 (Dezimalformat)
40018 = 25 (Dezimalformat)
40019 = 30 (Dezimalformat)

Automatisch konfigurierte Kommunikations-Ports

Der RS-485 Port

Bei einer Parent- oder Child-SPS muß der RS-485 (**exp link**) Port für die Verbindung zwischen den Einheiten der E/A-Erweiterungskopplung benutzt werden. In diesen beiden Betriebsarten sind die automatisch konfigurierten Portparameter festgelegt und können nicht verändert werden.

Bei einer einzelnen SPS kann der **exp link** Port nicht für E/A-Erweiterung benutzt werden. Er muß entweder für ASCII-Kommunikation eingesetzt oder disabled werden. Durch die automatische Konfiguration werden beim Single-Modus die Parameter dieses Ports auf ASCII-Kommunikation eingestellt.

☞ **Hinweis** Es kann nur ein Kommunikationsport für ASCII Kommunikationsfunktionen festgelegt werden.

Die RS-232 Ports

Die 110CPU512 und 110CPU612 Micro SPS haben zwei RS-232 Kommunikationsports, die Modelle 110CPU311 und 110CPU411 hingegen nur einen. Die durch die automatische Konfiguration zugewiesenen Portparameter sind sowohl vom SPS-Modell als auch von der Betriebsart abhängig.

Wenn die SPS in der Betriebsart Parent oder Child arbeitet, konfiguriert die Programmiergerätesoftware einen der RS-232 Ports automatisch in einem Modus, der die Kommunikation zwischen der SPS und entweder einem ASCII-Eingabe/Ausgabegerät oder einem *Modbus Master*gerät unterstützt.

☞ **Hinweis** Modbus ist das Protokoll, das die Kommunikationen der Kontaktlogikprogrammierung zwischen einem Programmiergerät und der SPS abwickelt. Das Programmiergerät wird dabei als das Modbus Mastergerät betrachtet und die SPS als Modbus Slave.

Für den Toggle-Modus zwischen Modbus und ASCII benutzt der RS 232-Port seine DSR-Verbindung, um der SPS mitzuteilen, ob ein ASCII-Gerät oder ein Modbus Mastergerät angeschlossen ist.

- □ Wenn ein ASCII-Gerät, z.B. ein Drucker oder ein Zeichen anzeigender Monitor an **comm 1** angeschlossen ist, wird die DSR-Verbindung inaktiv und der Port überträgt 8-Bit ASCII-Daten.
- □ Wenn ein Modbus Mastergerät, z.B. das HHP oder ein PC auf dem Modsoft Lite läuft, an **comm 1** angeschlossen ist, wird die DSR-Verbindung aktiviert und der Port überträgt 8-Bit RTU-Daten.

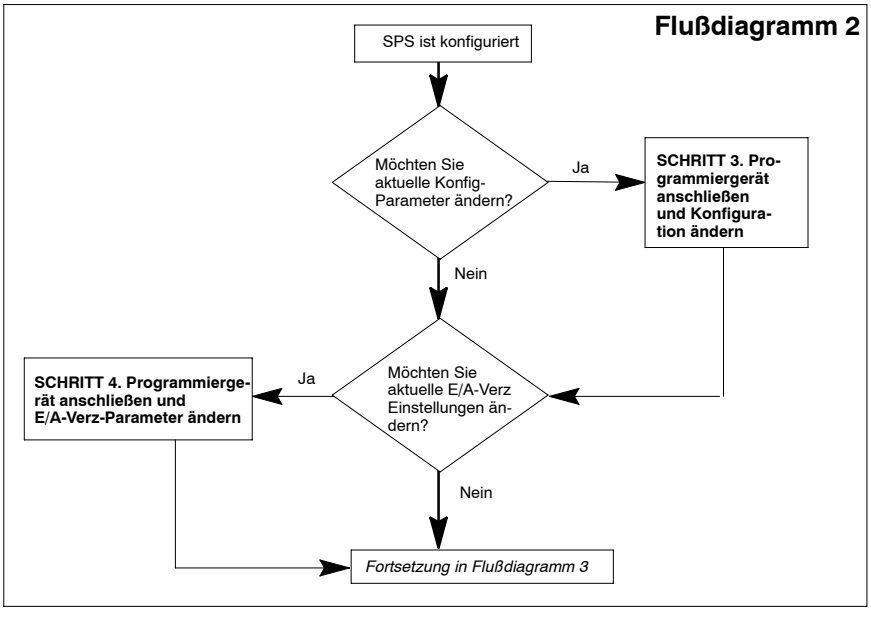
Wenn es sich bei der SPS um Modell 110CPU311 oder 110CPU411 in Parent- oder Childmodus handelt, wird bei der Autokonfiguration der Modbus/ASCII Toggle-Modus auf den **comm 1** Port gelegt. Wenn es sich bei der SPS um Modell 110CPU512 oder 110CPU612 handelt, wird bei der Autokonfiguration der Modbus/ASCII Toggle-Modus auf den **comm 2** Port gelegt und der **comm 1** Port ausschließlich für Modbus-Kommunikation konfiguriert.

Bei SPS im Single-Modus werden die RS-232 Ports immer automatisch als dezidierte *Modbus*-Kommunikationsports konfiguriert. Dies ist dadurch bedingt, daß der RS-485 Port automatisch für ASCII konfiguriert wird und nur ein Port eine SPS ASCII-Kommunikation unterstützen kann.

Alle RS-232 Ports werden automatisch für 9600 Baud konfiguriert, dadurch haben Sie die Möglichkeit, ein Programmiergerät an jedem beliebigen RS-232 Port der SPS anzuschließen. Geräte, die nicht mit dem Modbusprotokoll kommunizieren, können nicht an einen dezidierten Modbus Port angeschlossen werden.

☞ **Hinweis** Wenn Sie auf einem RS-232 Port mit 9600 Baud arbeiten, sollten auf dem anderen RS-232 Port 2400 Baud nicht überschritten werden.

Modifizierung der Konfigurationsparameter



Abhängig vom Programmiergerät, das Sie benutzen, können Sie eventuell viele der Autokonfigurationseinstellungen der SPS ändern. Das HHP erlaubt nur bei wenigen automatisch konfigurierten Parametern Änderungen, wohingegen MODSOFT Lite Ihnen sehr viel Flexibilität beim Festlegen der Konfiguration gibt.

Achtung Wenn Sie ein HHP benutzen, um Änderungen an einer bestehenden SPS-Konfiguration zu machen, werden die gesamte Kontaktplanlogik, die E/A-Verzeichnisse und gegenwärtig im SPS-Speicher befindliche ASCII Nachrichtendaten gelöscht.

Die Anzahl der Referenzen

MODSOFT Lite ermöglicht Ihnen, die Zusammensetzung der Referenzen in Ihrer Konfiguration zu ändern. Die Gesamtzahl der Register kann jedoch nicht erhöht werden.

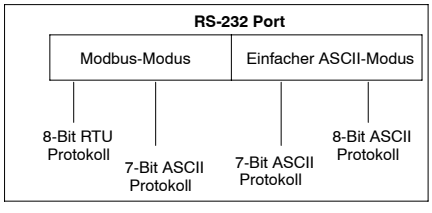
Wenn z.B. Ihre Anwendung 32 zusätzliche 0*x* Referenzen benötigt, können Sie 32 zur verfügbaren Gesamtsumme hinzufügen, wenn Sie die Anzahl der 1*x* Referenzen um 32 herabsetzen oder die Anzahl der 3*x* oder 4*x* Referenzen um 2 vermindern (3*x* und 4*x* Register enthalten 16 Bits, 0*x* und 1*x* Referenzen sind einzelne Bits).

Anzahl der Logiksegmente

Wenn Sie nicht vorhaben, Unterprogramme zu benutzen, sollte der durch die Autokonfiguration zugewiesene Wert von 2 nicht verändert werden. Das zweite Segment wird nie abgearbeitet, wenn es nicht durch einen JSR-Befehl oder einen festverdrahteten Interrupt aufgerufen wird.

RS-232 Kommunikationsparameter des RS-232 Port

Die RS-232 Ports können so eingestellt werden, daß sie entweder im Modbus- oder im einfachen ASCII-Modus arbeiten. Im Modbus-Modus fungiert der Port als Slave des Modbus-Mastergeräts, mit dem es verbunden ist. Im allgemeinen wird dies das Programmiergerät sein. Im einfachen ASCII-Modus wird dieser Port mithilfe der Kontaktplanlogik gelesen oder führt Schreiboperationen aus.(Im Kapitel 8 finden Sie eine Beschreibung der ASCII **COMM** Kontaktplanlogik-Befehle).



Modbus-Modus

Im Modbus-Modus kann der Port entweder mithilfe eines 8-Bit Remote Terminal Unit (RTU)-Protokolls oder eines 7-Bit ASCII-Protokolls kommunizieren. Auf dem **comm 1** RS-232 Port wird RTU nur mit 9600 Baud und ASCII nur mit 2400 Baud unterstützt. Der **comm 1** Port ist darüberhinaus auf gerade Parität und 1 Stopbit bei RTU und ASCII beschränkt. Die Modbus-Adressen des Ports können im Bereich 1 ... 247 liegen.

☞**Hinweis** Eine Micro SPS kann in einem Modbus Netzwerk einen Teilnehmer darstellen, wenn ihr eine eindeutige Modbus-Netzwerkadresse zugewiesen wird. Wenn sich die SPS nicht in einem Modbus-Netzwerk befindet, sollte die Adresse 1 bleiben. Befindet sich die SPS jedoch in einem Modbus-Netzwerk, muß ihre Adresse in Bezug auf alle anderen Teilnehmer des Netzwerkes im Bereich 1 ... 247 eindeutig sein. (Einzelheiten dazu finden Sie im ***Modicon Modbus Protocol Reference Guide***, PI-MBUS-300.)

Wenn Ihre SPS über einen **comm 2** RS-232 Port verfügt, stehen weitere optionale Parameter für den Port im Modbus-Modus zur Verfügung:

Optionale Comm Parameter für den comm 2 Port	
Baud	1200, 2400, 4800, 9600
Comm Modus	7-Bit ASCII, 8-Bit RTU
Paritätsprüfung	Ungerade, Gerade, Keine
Stopbits	1, 2
Modbus-Adresse	1 ... 247

Die zwei folgenden Kombinationen von RS-232 Portparametern werden auf den **comm1** oder **exp link** Ports für einfache ASCII-Kommunikation *nicht* unterstützt:

- ☐ 7-Bit ASCII mit 1 StopBit und keiner Parität
- ☐ 8-Bit ASCII mit 2 StopBits und gerader oder ungerader Parität

Einfacher ASCII-Modus

Im einfachen ASCII-Modus kann ein RS-232 Port nur mit einem ASCII-Protokoll kommunizieren, mit einer Auflösung von 7- oder 8-Bit. RTU-Kommunikationen sind im einfachen ASCII-Modus nicht zugelassen.

Ein RS-232 Port im einfachen ASCII-Modus kann mit jedem der folgenden Portparameter betrieben werden:

Optionale Comm Parameter für einfachen ASCII-Modus	
Baud	1200, 2400, 4800, 9600
Comm Modus	7-Bit ASCII, 8-Bit ASCII
Parität	Ungerade, Gerade, Keine
Stopbits	1, 2

Modem Kommunikationsmöglichkeiten

Der **comm 1** Port aller Modicon Micro SPS-Modelle ist mit einer Schaltung ausgestattet, die Modem Handshake-Signale unterstützt. Damit die Modemkommunikation möglich ist, muß sich der Port der SPS im dezidierten Modbus-Modus befinden und ein spezieller Adapter am Modemende der Kabelverbindung benutzt werden.

⚠ Achtung Aufgrund der speziellen Art, in der die DSR-Verbindung im Modbus/ASCII-Togglemodus arbeitet, kann in diesem Modus der comm 1 Port nicht über ein Modem kommunizieren.

Vier Adaptersätze sind von Modicon erhältlich, die die Teile enthalten, die Sie für einen Adapteranschluß Ihres Modems benötigen:

RJ45-zu-D-Shell Adapter Kits	
Adapter-Beschreibung	**Teilnummer**
RJ45-zu-9-Stift D-Shell, männlich	110XCA20301
RJ45-zu-9-Stift D-Shell, weiblich	110XCA20302
RJ45-zu-25-Stift D-Shell, männlich	110XCA20401
RJ45-zu-25-Stift D-Shell, weiblich	110XCA20402

comm 2 unterstützt keine Modemkommunikation.

Wenn Sie die Einheit für Modemkommunikation einrichten wollen, müssen Sie also **comm 1** als dezidierten Modbus-Port festlegen und seine Portparameter denen des Modems anpassen, z.B. 2400 Baud, ASCII-Modus.

Kommunikationsparameter des RS-485 Port

Wenn der **exp net** RS-485 Port als dedizierter ASCII-Port benutzt wird (das ist nur dann der Fall, wenn die SPS im Singlemodus arbeitet) stehen die folgenden Kommunikationsparameter zur Verfügung:

Optionale Comm Parameter für den exp net Port	
Baud	1200, 2400, 4800, 9600
Comm Modus	7-Bit ASCII, 8-Bit ASCII
Paritätsprüfung	Ungerade, Gerade, Keine
Stopbits	1, 2

Wenn der RS-485 Port für E/A-Erweiterung benutzt wird, d.h. wenn die SPS im Modus Parent oder Child arbeitet, sind die automatisch konfigurierten Portparameter vorgegeben und können nicht verändert werden.

Adressierung der E/A-Bereiche

Das E/A-Verzeichnis ist eine Tabelle im Speicher der SPS-Systemkonfiguration, die Referenznummern im SPS Anwender-Datenspeicher (0*x*, 1*x*, 3*x* und 4*x*) mit den realen Feldeingängen und -ausgängen koppelt.

Feste E/A-Bereiche

Im E/A-Verzeichnis Editor sind fünf lokale E/A-Bereiche für die Modicon Micro SPS reserviert.

- ☐ Bereich 1 für die Adressierung lokaler, diskreter Eingangs- und Ausgangsresourcen
- ☐ Bereich 2 für die Adressierung von Zähler-/Interrupteingängen
- ☐ Bereich 3 für die Adressierung von Timer-/Zählereingängen
- ☐ Bereich 4 für die Adressierung lokaler analoger Ein- und Ausgänge
- ☐ Bereich 5 für die Adressierung des Übertragungsregisters für einen *generalisierten Datentransfer* zwischen Parent- und Child-SPS

Nicht alle dieser Orte werden von allen SPS benutzt, z.B. der für analoge E/A reservierte Ort 4 nur in den Modellen 110CPU612. Wenn ein reservierter, lokaler E/A-Ort nicht benutzt wird, muß er im E/A-Verzeichnis leer gelassen werden. Er kann nicht für die Adressierung eines anderen Typs von Ein-/Ausgängen benutzt werden.

Wenn Sie auf Ihrem Programmiergerät das E/A-Verz betrachten, sind die Arten der E/A-Punkte jedes lokalen E/A-Bereiches durch einen alphanumerischen *Bereichstyp* bezeichnet. Die untenstehende Tabelle zeigt die standardmäßigen Ortsangaben für die lokalen Ressourcen aller Micro SPS-Modelle.

E/A-Verz Bereichstypen für lokale E/A

E/A-Ort	110CPU Model	Feste Ressourcen	Bereichstyp
Diskret (1)	31100, 41100 51200, 61200	16 ... 24 VDC Ein / 12 Relais Aus	MIC128
	31101, 41101, 51201	16 ... 115 VAC Ein / 8 Triac Aus 4 Relais Aus	MIC131
	31102, 41102, 51202	16 ... 230 VAC Ein / 8 Triac Aus 4 Relais Aus	MIC134
	31103, 41103, 51203, 61203	16 ... 24 VDC Ein / 12 FET Aus	MIC137
Zähler / Interrupt (2)	Alle 411, 512 und 612 Modelle	8-Bit Zähler /Interrupt Ein	MIC140
Timer / Zähler (3)	nicht verfügbar bei 311 Modellen	16-Bit Timer/aktueller Zählerwert	MIC147
Analog (4) Alle Ausgangskanäle haben 12-Bit Auflösung	nur 612 Modelle	4 Ein (0 ... 10, 12-Bit), 2 Aus	MIC141
		4 Ein (1 ... 5, 12-Bit), 2 Aus	MIC142
		4 Ein (± 10, 12-Bit), 2 Aus	MIC143
		4 Ein (0 ... 10, 15-Bit), 2 Aus	MIC144
		4 Ein (1 ... 5, 14-Bit), 2 Aus	MIC145
		4 Ein (± 10), 2 Aus	MIC146
Generalisierter Datentransfer (5)	Alle 512 und 612 Modelle	1 Wort Ein, 1 Wort Aus	MIC148
		2 Worte Ein, 2 Worte Aus	MIC149
		4 Worte Ein, 4 Worte Aus	MIC150
		8 Worte Ein, 8 Worte Aus	MIC151

Das Betriebssystem reserviert die ersten zwölf 0*x* Referenzen und die ersten sechzehn 1*x* Referenzen (00001 ... 00016 und 10001 ... 10016) für lokale

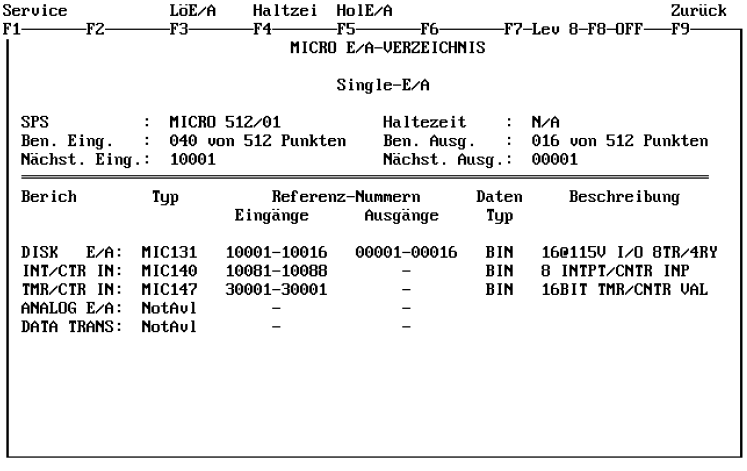
diskrete, auf der Baugruppe befindliche E/A.

Die lokalen Ressourcen einer 110CPU51201 SPS in der Betriebsart Single würden beispielsweise wie folgt adressiert:

- Bereichstyp **MIC131** für den ersten Bereich, um die diskreten E/A-Punkte festzulegen. Für die 115 VAC-Eingänge werden die Referenzen 10001 .. 10016 adressiert. Für die Triacausgänge werden die Referenzen 00001 ... 00008 und für die Relaisausgänge die Referenzen 00009 ... 00012 adressiert.
- Bereichstyp **MIC140** für den zweiten Bereich, um die Hochgeschwindigkeits-, Interrupt-/Zählereingänge festzulegen, für die die Referenzen 10081 ... 10088 adressiert werden.
- Bereichstyp **MIC147** für den dritten Bereich, um den Hochgeschwindigkeits-, Timer-/Zählereingang festzulegen, für den Register 30001 adressiert wird.

Die letzten zwei Register für analoge E/A und den generalisierten Datentransfer stehen in diesem E/A-Verz nicht zur Verfügung. Nur SPS vom Typ110CPU612 unterstützen analoge E/A und nur Parent- und Child-SPS unterstützen den gerneralisierten Datentransfer.

Nachstehend sehen Sie als Beispiel den Bildschirm eines E/A-Verzeichnisses von MODOSFT Lite, der die diskrete Adressierung verdeutlicht:


Service LöE/A Haltzei HolE/A Zurück
F1 F2 F3 F4 F5 F6 F7-Lev 8-F8-OFF F9
MICRO E/A-VERZEICHNIS
Single-E/A
SPS : MICRO 512/01 Haltezeit : N/A
Ben. Eing. : 040 von 512 Punkten Ben. Ausg. : 016 von 512 Punkten
Nächst. Eing.: 10001 Nächst. Ausg.: 00001
Berich Typ Referenz-Nummern Daten Beschreibung
Eingänge Ausgänge Typ
DISK E/A: MIC131 10001-10016 00001-00016 BIN 16@115V I/O 8TR/4RY
INT/CTR IN: MIC140 10081-10088 - BIN 8 INTPT/CNTR INP
TMR/CTR IN: MIC147 30001-30001 - BIN 16BIT TMR/CNTR VAL
ANALOG E/A: NotAvl - -
DATA TRANS: NotAvl - -

Adressierung der A120 E/A

Bei den Modellen 110CPU512 oder 110CPU612 besteht die Möglichkeit, eine optionale A120 E/A-Erweiterung zu benutzen. In diesem Fall muß die A120 E/A in den System-Konfigurationsspeicher der betreffenden SPS aufgenommen werden.

Sie müssen das E/A-Verzeichnis mit Hilfe der Programmiergeräte-Software editieren, um die A120 E/A zu adressieren. Jedem A120 E/A-Modul wird ein Ort in dem Baugruppenträger zugewiesen, der das Modul enthält.

Jeder physikalische Baugruppenträger, der mit der SPS verbunden ist — *Baugruppenträger #2*, *#3* und *#4* — kann bis zu fünf E/A-Bereiche enthalten. Bis zu 20 A120 E/A-Module (oder Bereiche) können in dem E/A-Verzeichnis einer Micro SPS adressiert werden. In der SPS sind die folgenden Referenzen für erweiterte E/A-Adressierung reserviert:

- ☐ Referenzen 00017 ... 00080 für die Adressierung diskreter A120 Ausgangspunkte
- ☐ Referenzen 10017 ... 10080 für die Adressierung diskreter A120 Eingangspunkte
- ☐ Referenzen 30002 ... 30005 und 30011 ... 30030 für die Adressierung von Register-/Anlalogeingängen der A120 E/A
- ☐ Die Referenzen 40003 ... 40010 sind für die Zuweisung von Registerausgängen der A120 E/A reserviert.

  Hinweis Wenn sie nicht für die A120 E/A-Adressierung benötigt werden, dürfen diese reservierten Referenzen für die Adressierung von lokalen E/A-Ressourcen in anderen Steuerungen innerhalb der E/A-Erweiterungskopplung benutzt werden.

Beispiel: Eine Micro SPS mit einem A120 E/A-Baugruppenträger

Das folgende Beispiel enthält zwei MODSOFT Lite E/A-Verzeichnis-Bildschirme. Das System, dessen E/A zugewiesen wird, besteht aus einer 110CPU51200 SPS und einem Baugruppenträger mit fünf A120 E/A-Modulen — zwei BDAP212 und drei BDAP216.

Bei dieser Anwendung benutzt die SPS nur einen ihrer diskreten E/A-Punkte. Daher werden insgesamt in dieser Konfiguration sechs E/A-Bereiche benutzt=MIC128 für die lokalen E/A-Punkte und fünf Bereiche für die A120 E/A-Module.

Bildschirm 1 zeigt das E/A-Verzeichnis für die lokalen Ressourcen der 110CPU51200 SPS. Diese SPS wird aufgrund der A120 E/A-Erweiterung als Baugruppenträger #1 betrachtet. Beachten Sie, daß in Baugruppenträger #1 nur die Orte 1, 2 und 3 zugänglich sind.

☞ **Hinweis** Bei MODSOFT Lite wird die E/A jedes Baugruppenträgers auf einem eigenen Bildschirm dargestellt. Zwischen diesen Bildschirmen, d.h. den Baugruppenträgern, können Sie sich durch Betätigen der Tasten <**PgUp**> und <**PgDn**> vorwärts oder rückwärts bewegen.

Die A120 E/A im Baugruppenträger 2 wird im E/A-Verzeichnis auf Bildschirm 2 gezeigt. Die A120 Eingangspunkte wurden den Referenzen 10033 ... 10056 und die Ausgangspunkte den Referenzen 00017 ... 00096 im Anwender-Datenspeicher der SPS zugewiesen.

Diese Anwendung benutzt insgesamt 56 diskrete Eingänge, 80 diskrete Ausgänge

sowie einen
Zähler/Timer-Registereingang.

```
Service          LöE/A    Haltzei  HolE/A                                Zurück
F1———————F2———————F3———————F4———————F5———————F6———————F7-Lev 8-F8-OFF——F9———————
                          MICRO E/A-VERZEICHNIS

                                  Parent-E/A

SPS          :  MICRO 512/00           Haltezeit    :  N/A
Ben. Eing.   :  040 von 512 Punkten    Ben. Ausg.   :  016 von 512 Punkten
Nächst. Eing.:  10001                  Nächst. Ausg.:  00001
═══════════════════════════════════════════════════════════════════════════
Berich        Typ          Referenz-Nummern      Daten     Beschreibung
                         Eingänge      Ausgänge    Typ

DISK   E/A:  MIC128   10001-10016   00001-00016   BIN   16@24V  I/O 12 RY
INT/CTR IN:  MIC140   10081-10088         -       BIN   8 INTPT/CNTR INP
TMR/CTR IN:  MIC147   30001-30001         -       BIN   16BIT TMR/CNTR VAL
ANALOG E/A:  NotAvl         -               -
DATA TRANS:  NotAvl         -               -
```

Bildschirm 1. E/A-Verzeichnis für die lokalen E/A-Bereiche (Baugruppenträger 1)

```
Service          DropLö   HolE/A                                      Zurück
F1———————F2———————F3———————F4———————F5———————F6———————F7-Lev 8-F8-OFF——F9———————
                          E/A-VERZEICHNIS
                       A120 ERWEITERUNGS-E/AEN

   SPS          :     MICRO 512/00   Rack          :            2
   Anzahl Eing. :     56             Anzahl Ausg.  :            64

  ═══════════════════════════════════════════════════════════════════════
  Steck-  Modul-      Referenz-Nummern           Daten-   Modul-
  platz   Typ       Eingang       Ausgang        Typ      Beschreibung

   201    DAP212 10001 -10008   00001 -00008          Help Alt-H
   202    DAP212 10025 -10032   00025 -00032          Help Alt-H
   203    DAP216                00049 -00064          16-O 24V
   204    DAP216                00065 -00080          16-O 24V
   205
```

Bildschirm 2. E/A-Verzeichnis für A120 E/A-Bereiche (Baugruppenträger 2)

Adressierung der E/A in einer Erweiterungskopplung

Eine E/A-Erweiterungskopplung wird durch eine Kettenschaltung (daisy chain) von bis zu fünf Micro SPS mit Hilfe der Kabelverbindungen an den RS-485 Ports hergestellt. Eine SPS muß dabei als Parent, alle übrigen SPS müssen als children konfiguriert werden.

Die Parent-SPS

Die Parent-SPS kann ihre gesamten, eigenen E/A-Ressourcen adressieren wie auch beliebige lokale E/A-Ressourcen, die sich in den Child-Steuerungen befinden.

Die lokalen E/A-Bereiche der Parent-SPS werden automatisch adressiert. Referenzen für die Zuordnung zusätzlicher E/A-Punkte der Parent-SPS sind wie folgt verfügbar:

Physikalische Eingänge	Referenzen (im Anwender-Datenspeicher)	Physikalische Ausgänge
	00001 ... 00012	Lokale lokale diskrete Ausgänge (12)
	00017 ... 00080	Reserviert (diskret, A120 oder Child)
	00081	Batterie OK Spule
Lokale lokale diskrete Eingänge (16)	10001 ... 10016	
Reserviert (diskret, A120 oder Child)	10017 ... 10080	
Lokale Interrupt/Zähler Eingänge (8)	10081 ... 10088	
Reserviert (Child Interrupt/Timer)	10089 ... 10120	
Lokaler Timer/ Zähler Eingang (1)	30001	
Reserviert (Child Timer/Zähler)	30002 ... 30005	
Lokale lokale analoge Eingänge (4)	30006 ... 30010	
Reserviert (Child analoge Eingänge)	30011 ... 30030	
	40001 ... 40002	Lokale lokale analoge Ausgänge (2)
	40003 ... 40010	Reserviert (Child, analoge Ausgänge)
	40011	10 ms Timer
	40012 ... 40019	Uhr

Eine Child-SPS

Wenn Sie für eine SPS die Betriebsart Child wählen, nimmt das Kontaktplan-Betriebssysten standardmäßig an, daß alle lokalen E/A-Punkte dieser SPS von der Parent-SPS in diesem Netzwerk gesteuert werden. Gemäß dieser Voreinstellung werden daher dem E/A-Verzeichnis einer Child-SPS keine Werte zugewiesen.

Die lokalen E/A-Bereiche des Child können in einem Bildschirm, der mit dem E/A-Verzeichnis des Parent verbunden ist, zugewiesen werden.

☞ **Hinweis** Jede A120 E/A, die mit einer Child verbunden ist, muß von diesem Child adressiert werden. Auf die A120 E/A in einem Child kann nicht vom Parent in dieser E/A-Erweiterungskopplung zugegriffen werden, sie kann auch nicht vom Parent gesteuert werden.

Beispiel: Eine Erweiterungskopplung, bei der die gesamte, lokale E/A vom Parent gesteuert wird

Das im folgenden Beispiel konfigurierte System besteht aus zwei 110CPU51200 SPS, einer Parent und einer Child in einer E/A-Erweiterungskopplung. Das Beispiel zeigt drei MODSOFT Lite E/A-Verzeichnis Bildschirme.

Stellen Sie bei der Konfiguration der Parent sicher, daß diese für mindestens ein Child eingerichtet wird. Das Betriebssystem läßt nur dann einen Zugriff des Parent auf die lokale E/A-Ressourcen des Child zu, wenn die Existenz dieses Child in der Konfiguration des Parent festgelegt wurde.

Die Bildschirme 1 und 2 zeigen die E/A-Verzeichnisse für die lokalen E/A-Bereiche der Parent und des Child, die von Referenzen im Speicher des Parent gesteuert werden. Beide E/A-Verzeichnis Bildschirme sind zugänglich, wenn das Programmiergerät an den Parent angeschlossen ist.

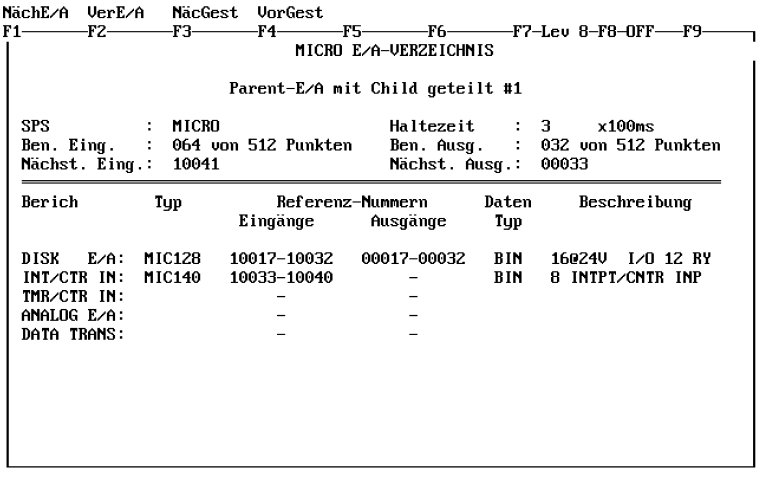
```
Service          LöE/A   Haltzei  HolE/A                              Zurück
F1------F2------F3------F4------F5------F6------F7-Lev 8-F8-OFF--F9------
                          MICRO E/A-VERZEICHNIS

                                Parent-E/A

SPS          :  MICRO 512/00          Haltezeit    :  N/A
Ben. Eing.   :  040 von 512 Punkten   Ben. Ausg.   :  016 von 512 Punkten
Nächst. Eing.:  30002                 Nächst. Ausg.:  00017
═══════════════════════════════════════════════════════════════════════════
Berich        Typ          Referenz-Nummern      Daten     Beschreibung
                         Eingänge      Ausgänge    Typ

DISK   E/A:  MIC128   10001-10016   00001-00016   BIN   16@24V  I/O 12 RY
INT/CTR IN:  MIC140   10081-10088         -       BIN   8 INTPT/CNTR INP
TMR/CTR IN:  MIC147   30001-30001         -       BIN   16BIT TMR/CNTR VAL
ANALOG E/A:  NotAvl         -              -
DATA TRANS:  NotAvl         -              -
```

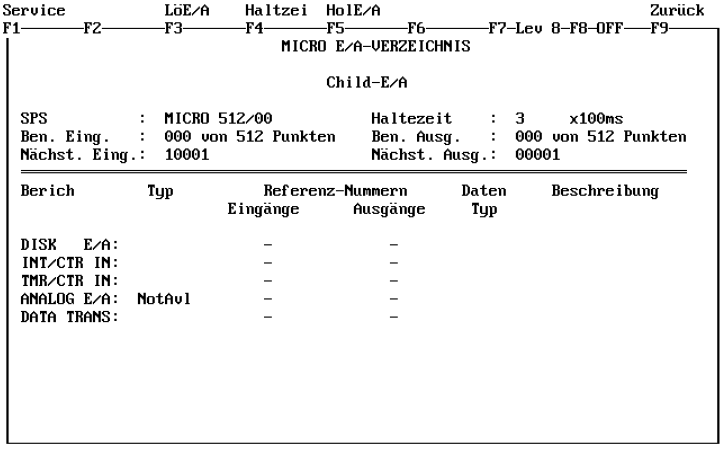
Bildschirm 1. E/A-Verzeichnis für die lokalen E/A-Punkte des Parent


NächE/A VerE/A NäcGest VorGest
F1 F2 F3 F4 F5 F6 F7-Lev 8-F8-OFF F9
MICRO E/A-VERZEICHNIS
Parent-E/A mit Child geteilt #1
SPS : MICRO
Haltezeit : 3 x100ms
Ben. Eing. : 064 von 512 Punkten
Ben. Ausg. : 032 von 512 Punkten
Nächst. Eing.: 10041
Nächst. Ausg.: 00033
Berich Typ Referenz-Nummern Daten Beschreibung
Eingänge Ausgänge Typ
DISK E/A: MIC128 10017-10032 00017-00032 BIN 16@24V I/O 12 RY
INT/CTR IN: MIC140 10033-10040 - BIN 8 INTPT/CNTR INP
TMR/CTR IN: - -
ANALOG E/A: - -
DATA TRANS: - -

Bildschirm 2. E/A-Verzeichnis für die lokalen E/A-Punkte des Child, auf die von dem Parent zugegriffen wird

Beachten Sie, daß die im E/A-Verzeichnis für das Child benutzten Bereichssangaben die gesamten Orte der lokalen, diskreten Eingänge und Relaisausgänge der Steuerung durch die Parent unterstehen. `MIC128` weist alle 16 24VDC-Eingänge des Child den Referenzen 10017 ... 10032 im Speicher des Parent zu sowie alle 12 Relaisausgänge den Referenzen 10033 ... 10040 dem Anwender-Datenspeicher des Parent, `MIC140` weist die Hochgeschwindigkeits-Eingänge den Referenzen 10097 ... 10104 des Anwender-Datenspeichers des Parent zu.

Als Ergebnis enthält der E/A-Verzeichnis Bildschirm, der beim Anschluß des Programmiergeräts an das Child erscheint (vgl. Bilschirm 3 unten), keine Bereichstypen:


Service LöE/A Haltzei HolE/A Zurück
F1 F2 F3 F4 F5 F6 F7-Lev 8-F8-OFF F9
MICRO E/A-VERZEICHNIS
Child-E/A
SPS : MICRO 512/00
Haltezeit : 3 x100ms
Ben. Eing. : 000 von 512 Punkten
Ben. Ausg. : 000 von 512 Punkten
Nächst. Eing.: 10001
Nächst. Ausg.: 00001
Berich Typ Referenz-Nummern Daten Beschreibung
Eingänge Ausgänge Typ
DISK E/A: - -
INT/CTR IN: - -
TMR/CTR IN: - -
ANALOG E/A: NotAvl - -
DATA TRANS: - -

Bildschirm 3. E/A-Verzeichnis für die lokalen E/A-Punkte des Child

Aufteilung lokaler E/A zwischen Parent- und den Child-SPS

Eine Child-SPS hat die Möglichkeit, ihre lokalen E/A-Ressourcen teilweise an die Parent abzutreten - d.h. die Child behält die Kontrolle über einen Teil ihrer lokalen E/A und die Parent kontrolliert die restliche E/A des Child. Wenn lokale E/A-Ressourcen aufgeteilt werden, müssen die von dem Child gesteuerten E/A-Ressourcen über das E/A-Verzeichnis des Child und die von dem Parent gesteuerten E/A-Punkte vom E/A-Verzeichnis des Parent adressiert werden.

Der Schlüssel für eine Aufteilung der E/A liegt in der Wahl der richtigen Bereichsangabe (vgl. unstenstehende Tabelle) und der Zuweisung in den E/A-Verzeichnis Bildschirmen des Parent und des Child.

Wenn z.B. ein Child 12 lokale FET-Ausgänge hat, können Sie die E/A im E/A Verzeichnis einer SPS mit einer Bereichsangabe vom Typ `MIC138` (wobei Sie ihr 8 FET-Ausgänge unterordnen), und das andere E/A-Verzeichnis mit einer Bereichsangabe vom Typ `MIC139` definieren (wobei Sie die restlichen 4 FET-Ausgänge der Kontrolle der anderen SPS unterstellen).

E/A Verzeichnis der Bereichstypen für lokale E/A

E/A-Typ		Bereich-styp	110CPU Modelle
Diskret	16 ... 24 VDC Ein / 12 Relais Aus	MIC128	31100, 41100, 51200, 61200
	16 ... 24 VDC Ein / 8 Relais Aus	MIC129	
	16 ... 24 VDC Ein / 4 Relais Aus	MIC130	
	16 ... 115 VAC Ein / 8 Triac Aus 4 Relais Aus	MIC131	31101, 41101, 51201
	16 ... 115 VAC Ein / 8 Triac Aus	MIC132	
	16 ... 115 VAC Ein / 4 Relais Aus	MIC133	
	16 ... 230 VAC Ein / 8 Triac Aus 4 Relais Aus	MIC134	31102, 41102, 51202
	16 ... 230 VAC Ein / 8 Triac Aus	MIC135	
	16 ... 230 VAC Ein / 4 Relais Aus	MIC136	
	16 ... 24 VDC Ein / 12 FET Aus	MIC137	31103, 41103, 51203, 61203
	16 ... 24 VDC Ein / 8 FET Aus	MIC138	
	16 ... 24 VDC Ein / 4 FET Aus	MIC139	
Zähler / Interrupt	8-Bit Zähler/Interrupt Ein	MIC140	All 311, 512, & 612 Modelle
Analog (nur für 612 Modelle) Alle Ausgangskanäle haben 12-Bit Auflösung	4 Ein (0 ... 10, 12-Bit), 2 Aus	MIC141	61200, 61203
	4 Ein (1 ... 5, 12-Bit), 2 Aus	MIC142	
	4 Ein (± 10, 12-Bit), 2 Aus	MIC143	
	4 Ein (0 ... 10, 15-Bit), 2 Aus	MIC144	
	4 Ein (1 ... 5, 14-Bit), 2 Aus	MIC145	
	4 Ein (± 10), 2 Aus	MIC146	
Timer / Zähler	16-Bit Timer/aktueller Zählerstand	MIC147	Standardeinstellung bei allen Modellen ist KEINE
Generalisierter Datentransfer	1 Wort Ein, 1 Wort Aus	MIC148	
	2 Worte Ein, 2 Worte Aus	MIC149	
	4 Worte Ein, 4 Worte Aus	MIC150	
	8 Worte Ein, 8 Worte Aus	MIC151	

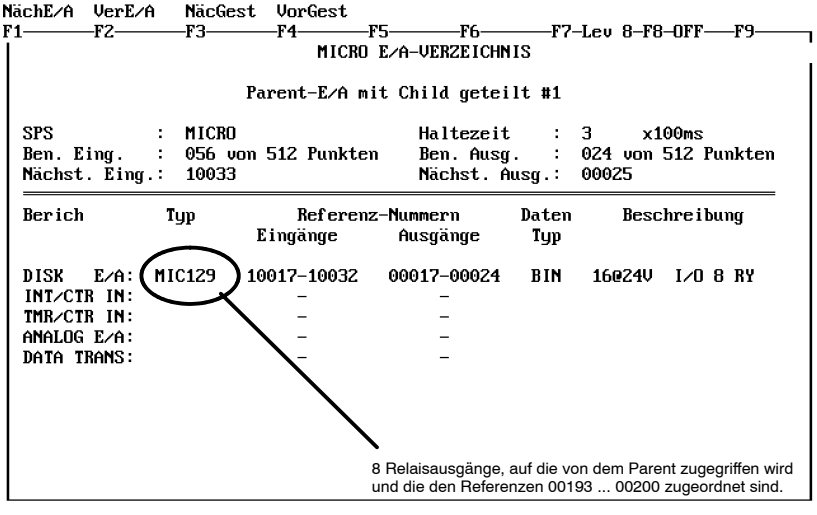
Beide SPS lesen die gleichen Eingangsdaten. Gemeinsame Eingangsdaten führen nicht zu Konflikten zwischen Parent und Child, daher können die gleichen, lokalen *Eingänge* sowohl dem Parent als auch dem Child zugewiesen werden.

Wenn jedoch beide SPS die gleichen Ausgangsdaten schreiben, können Fehler entstehen. Werden die gleichen Ausgänge beiden SPS zugewiesen, wird das System einen Fehler bei dem Parent feststellen und dies in der SPS Statustabelle mit *Fehler* markieren .

Beispiel: Aufgeteilte E/A

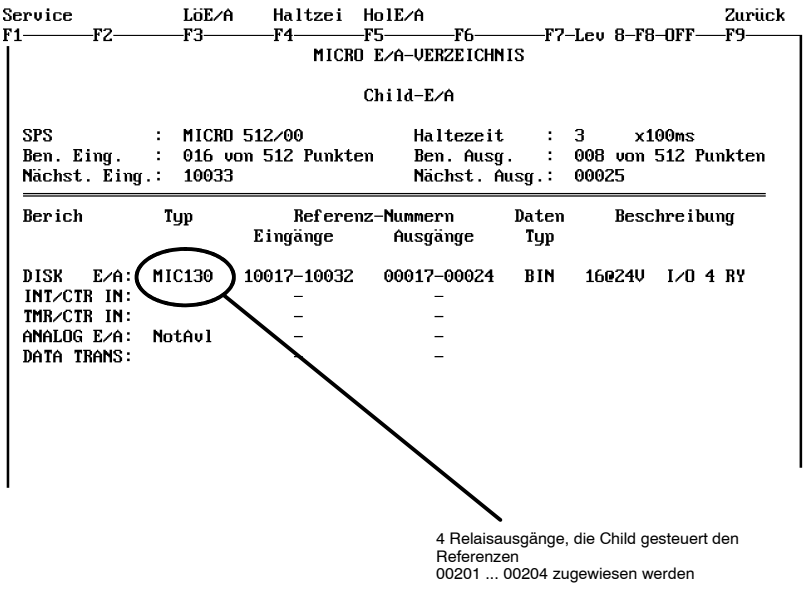
Das folgende Beispiel zeigt zwei MODSOFT Lite E/A-Verzeichnis Bildschirme. Sie zeigen, wie die 12 lokalen Relaisausgänge einer 110CPU51200 SPS, die als Child konfiguriert wurde, zwischen ihm und ihrem Parent aufgeteilt werden können.

Der untenstehende Bildschirm 1 zeigt das Verzeichnis der Child-E/A, auf die von dem Parent zugegriffen wird. Dieser E/A-Verzeichnis Bildschirm wird erstellt, wenn das Programmiergerät an die Parent-SPS angeschlossen ist. Der Bereichstyp für die diskrete E/A ist `MIC129`, wodurch festgelegt ist, daß die Parent auf acht der lokalen Relaisausgänge des Child Zugriff hat.


NächE/A VerE/A NäcGest VorGest
F1 F2 F3 F4 F5 F6 F7-Lev 8-F8-OFF F9
MICRO E/A-VERZEICHNIS
Parent-E/A mit Child geteilt #1
SPS : MICRO Haltezeit : 3 x100ms
Ben. Eing. : 056 von 512 Punkten Ben. Ausg. : 024 von 512 Punkten
Nächst. Eing.: 10033 Nächst. Ausg.: 00025
Berich Typ Referenz-Nummern Daten Beschreibung
Eingänge Ausgänge Typ
DISK E/A: MIC129 10017-10032 00017-00024 BIN 16@24V I/O 8 RY
INT/CTR IN: - -
TMR/CTR IN: - -
ANALOG E/A: - -
DATA TRANS: - -
8 Relaisausgänge, auf die von dem Parent zugegriffen wird
und die den Referenzen 00193 ... 00200 zugeordnet sind.

Bildschirm 1: Child E/A, auf die von dem Parent zugegriffen wird

Bildschirm 2 stellt das Verzeichnis der Child E/A dar, die der Kontrolle des Child verbleibt. Dieser E/A-Bildschirm wird erstellt, wenn das Programmiergerät an die Child-SPS angeschlossen ist. Der Bereichstyp für die diskrete E/A ist `MIC130`, wodurch festgelegt ist, daß das Child auf vier ihrer lokalen Relaisausgänge Zugriff hat.


Service LöE/A Haltzei HolE/A Zurück
F1 F2 F3 F4 F5 F6 F7-Lev 8-F8-OFF F9
MICRO E/A-VERZEICHNIS
Child-E/A
SPS : MICRO 512/00 Haltezeit : 3 x100ms
Ben. Eing. : 016 von 512 Punkten Ben. Ausg. : 008 von 512 Punkten
Nächst. Eing.: 10033 Nächst. Ausg.: 00025
Berich Typ Referenz-Nummern Daten Beschreibung
Eingänge Ausgänge Typ
DISK E/A: MIC130 10017-10032 00017-00024 BIN 16@24V I/O 4 RY
INT/CTR IN: - -
TMR/CTR IN: - -
ANALOG E/A: NotAvl - -
DATA TRANS: - -
4 Relaisausgänge, die Child gesteuert den Referenzen
00201 ... 00204 zugewiesen werden

Bildschirm 2: lokale E/A-Ressourcen, die von dem Child gesteuert werden

Generalisierter Datentransfer

Grundsätzlich ist die E/A-Erweiterungskopplung eine Möglichkeit, systemweit auf E/A-Ressourcen für die Abarbeitung eines Logikprogramms zuzugreifen, das in einer einzelnen SPS, dem Parent, läuft. Da jedoch jede einzelne Child-SPS in der Kopplung die Möglichkeit hat, ihre eigene Anwenderlogik zu speichern und ihre eigenen E/A-Kommunikationsports zu verwalten, kann bis zu einem gewissen Grad in verschiedenen CPUs der Kopplung eine *gleichzeitige Verarbeitung* (coprocessing) stattfinden.

Generalsierter Datentransfer ist ein Werkzeug, das es dem Parent- und den Child-SPS innerhalb der Kopplung ermöglicht, Daten auszutauschen, die keine Steuerdaten sind. Um Daten zwischen den Einheiten zu senden und zu empfangen, werden die Kabelverbindungen der Erweiterungskopplung benutzt.

Die Parent kann mit jedem beliebigen Child oder allen Children generalsierte Daten austauschen, eine Child kann nur mit dem Parent generalsierte Daten austauschen.

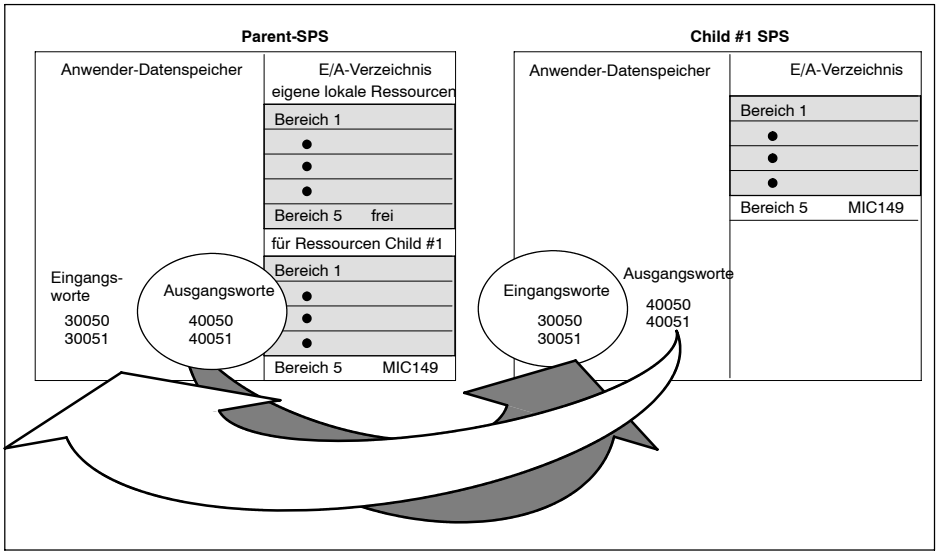
Die Parent und die Child-SPS eines E/A-Erweiterungsnetzes können bidirektional eine bestimmte Anzahl von Worten, die keine Steuerdaten sind, über das E/A-Erweiterungs-Netzwerk austauschen. Der 5. E/A-Bereich (DATA TRANS) ist bei allen Modicon Micro SPS für diesen generalisierten Datentransfer reserviert.

Sie können sowohl die Eingangs- als auch Ausgangsworte wählen, die im Anwender-Datenspeicher dem Parent und der Töchter SPS reserviert werden sollen, indem Sie eine der folgenden Ortsangaben in den E/A-Verzeichnissen vom Parent und Child spezifizieren:

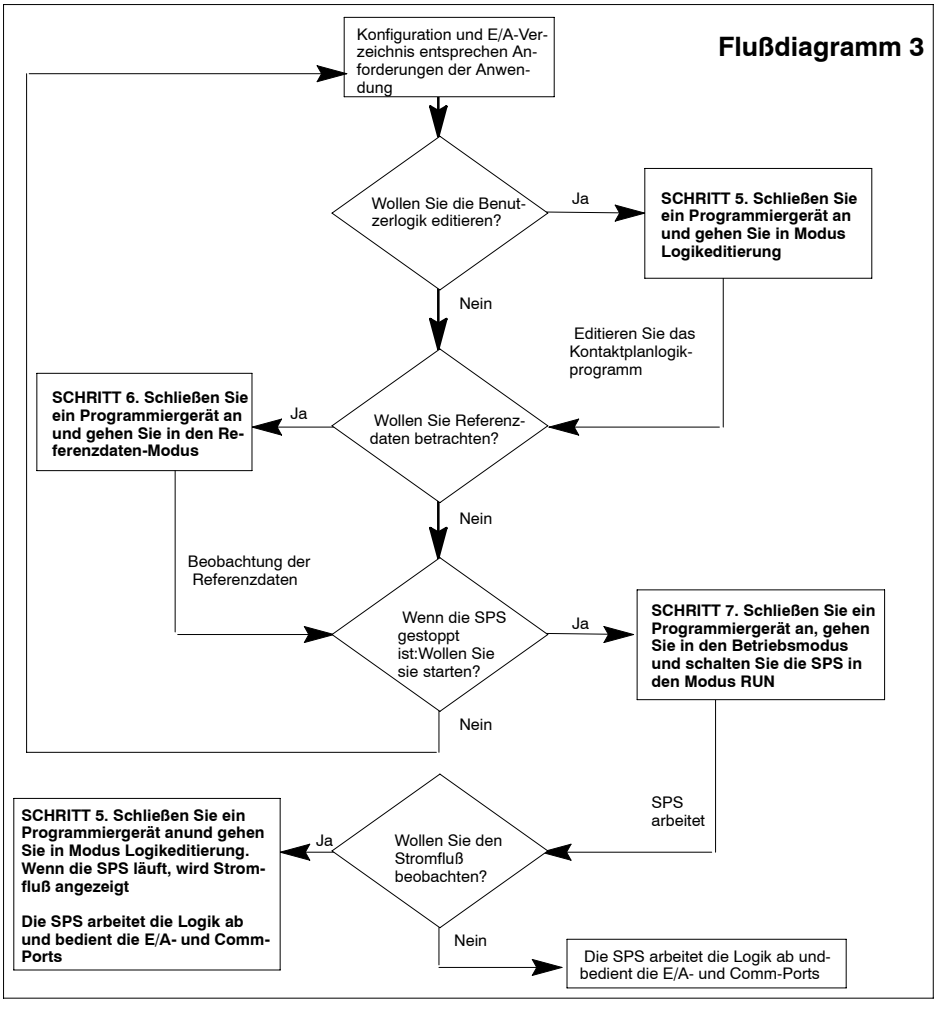
- ☐ **`MIC148`** spezifiziert ein Eingangswort (von 1*x* oder 3*x* Referenzen) und ein Ausgangswort (von 0*x* oder 4*x* Referenzen)
- ☐ **`MIC149`** spezifiziert zwei Eingangsworte (von 1*x* oder 3*x* Referenzen) und 2 Ausgangsworte (von 0*x* oder 4*x* Referenzen)
- ☐ **`MIC150`** spezifiziert vier Eingangsworte (von 1*x* oder 3*x* Referenzen) und vier Ausgangsworte (von 0*x* oder 4*x* Referenzen)
- ☐ **`MIC151`** spezifiziert acht Eingangsworte (von1*x* oder 3*x* Referenzen) und acht Ausgangsworte (von 0*x* oder 4*x* Referenzen)

Für die Einrichtung eines generalsierten Datentransfers zwischen einer Parent und einer Child-SPS müssen die gleichen Bereichsangaben in den E/A-Verzeichnissen des Child und des Parent gemacht werden. Wenn das Programmiergerät an die Parent-SPS angeschlossen ist, müssen Sie den Bereichstyp für den generalisierten Datentransfer in das E/A-Verzeichnis eingeben, das die lokalen E/A-Ressourcen des Child definiert, nicht in das E/A-Verzeichnis, das die lokalen E/A-Ressourcen des Parent festlegt.

Es folgt eine Illustration des Transferprozesses bei generalisierten Daten:



SPS Operationen


Flußdiagramm 3
Konfiguration und E/A-Verzeichnis entsprechen Anforderungen der Anwendung
Wollen Sie die Benutzerlogik editieren?
Ja
SCHRITT 5. Schließen Sie ein Programmiergerät an und gehen Sie in Modus Logikeditierung
Nein
Editieren Sie das Kontaktplanlogikprogramm
SCHRITT 6. Schließen Sie ein Programmiergerät an und gehen Sie in den Referenzdaten-Modus
Ja
Wollen Sie Referenzdaten betrachten?
Nein
Beobachtung der Referenzdaten
Wenn die SPS gestoppt ist:Wollen Sie sie starten?
Ja
SCHRITT 7. Schließen Sie ein Programmiergerät an, gehen Sie in den Betriebsmodus und schalten Sie die SPS in den Modus RUN
Nein
SPS arbeitet
SCHRITT 5. Schließen Sie ein Programmiergerät anund gehen Sie in Modus Logikeditierung. Wenn die SPS läuft, wird Stromfluß angezeigt
Die SPS arbeitet die Logik ab und bedient die E/A- und Comm-Ports
Ja
Wollen Sie den Stromfluß beobachten?
Nein
Die SPS arbeitet die Logik ab und-bedient die E/A- und Comm-Ports

Wenn einmal die SPS für die gewünschte Betriebsart konfiguriert wurde und die E/A-Bereiche in dem E/A-Verzeichnis adressiert worden sind, können Sie:

- ☐ Ihr Kontaktplanlogik-Programm erstellen oder editieren
- ☐ Referenzdaten beobachten und editieren
- ☐ die SPS starten und stoppen
- ☐ in einem laufenden Anwendungsprogramm den Stromfluß beobachten

In den nächsten Kapiteln werden wir den Befehlsvorrat der Kontaktplanlogik genauer betrachten und den Gebrauch von Befehlen für die Erstellung von Anwendungsprogrammen besprechen.

Kapitel 3
Grundlagen der Kontaktplanlogik-Programmierung

- ☐ Segmente und Netzwerke
- ☐ Standardelemente der Kontaktplanlogik
- ☐ Anwendungsbeispiel : Start-/Stopschaltung eines Motors
- ☐ Standardbefehle der Modicon Micro SPS
- ☐ Befehle für bestimmte Modelle der Modicon Micro SPS

Segmente und Netzwerke

Kontaktplanlogik-Segmente

Die gesamte Kontaktplanlogik für die Steuerung Ihrer Anwendung ist in einem Logik*segment* im Anwenderspeicher gespeichert. Wenn Sie Unterprogramme aufrufen, die Teil Ihrer Anwendung sind, müssen diese sich in einem separaten Segment befinden.

Die Modicon Micro SPS stellen eine Konfiguration mit zwei Segmenten zur Verfügung. Segment 1 beinhaltet die normale Kontaktplanlogik für die Steuerung der Anwendung. In Segment 2 befinden sich alle Unterprogramme. Die Logik der Unterprogramme wird nur dann abgearbeitet, wenn sie aufgerufen wird, was entweder von der Kontaktplanlogik oder einem externen Ereignis, das einen Interrupt auslöst, geschehen kann. Als Teil der regulären Logik wird Segment 2 daher nicht abgearbeitet.

Kontaktplan-Netzwerke

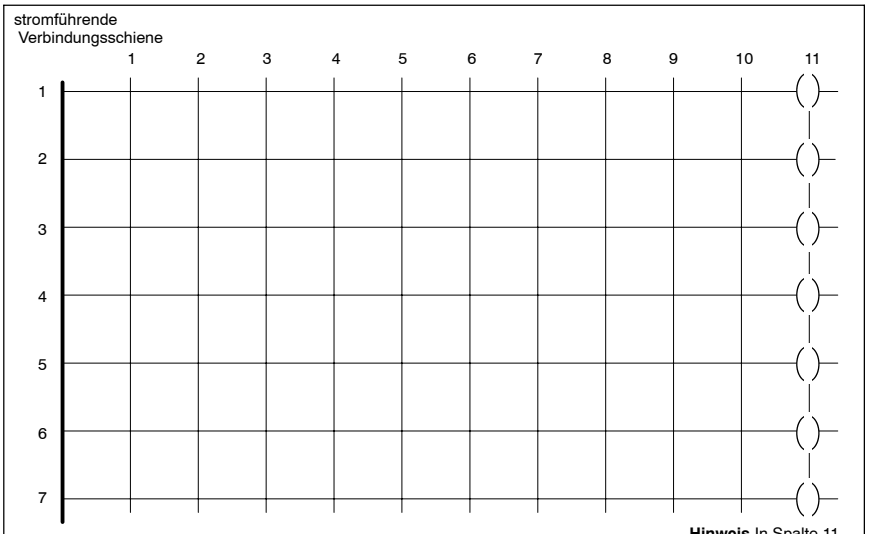
Jedes Segment besteht aus einer Gruppe aufeinanderfolgender *Netzwerke*. Jedes Netzwerk ist ein kleines, klar definiertes Kontaktplandiagramm, das links von einer stromführenden Verbindung begrenzt wird und rechts ebenfalls von einer stromführenden Verbindung begrenzt wird, die gemäß den Konventionen jedoch nicht angezeigt wird. Der Kontaktplan ist sieben Strompfade hoch und elf Spalten breit.

Die Schnittstelle jedes Strompfads mit einer Spalte des Netzwerks wird Element genannt — jedes Netzwerk enthält 77 Elemente.

Es gibt keine vorgeschriebene Begrenzung für die Anzahl der Netzwerke, die in einem Segment enthalten sein können — die Gesamtprogrammgröße ist durch den verfügbaren *Anwender-Datenspeicher* der CPU begrenzt, sowie durch die Zykluszeit, die die CPU für die Abarbeitung des Kontaktplanlogik-Programms benötigt.

Einsetzen von Schaltlogik und Befehlen in einem Netzwerk

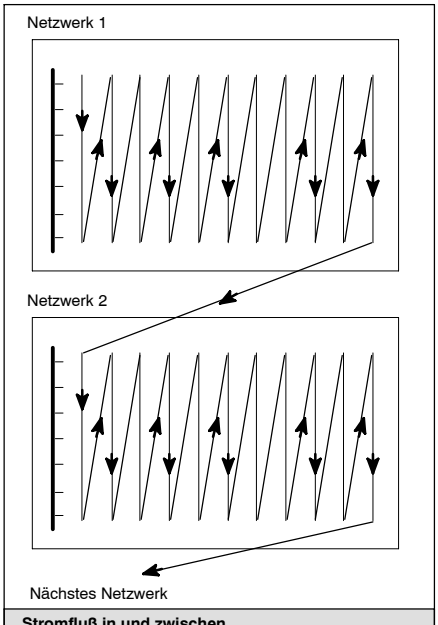
Jedes Mal, wenn Sie ein *Schaltlogikelement* benutzen — d.h. einen Kontakt, eine Spule, eine vertikale oder horizontale Verbindung — wird dadurch ein Element des Logik-Netzwerks belegt.



Kontaktplanlogik-Netzwerkstruktur

Hinweis In Spalte 11 können nur Spulen dargestellt werden

Abhängig vom Befehlstyp kann in der Kontaktplanlogik ein *Befehl* ein, zwei oder drei Elemente eines Netzwerks belegen. Ein Zählerbefehl ist z.B. ein Zweierelement-Befehl — er belegt zwei aufeinanderfolgende Elemente, die direkt übereinander liegen müssen. Ein ADD-Befehl hingegen ist ein Dreierelement-Befehl, der drei aufeinanderfolgende, direkt überanderliegende Elemente benötigt.



Stromfluß in und zwischen Kontakplanlogik-Netzwerken

Wie Kontaktplanlogik abgearbeitet wird

Eine Modicon Micro SPS bearbeitet die Kontaktplanlogik in der folgenden Reihenfolge:

- ☐ Segment auf Segment
- ☐ Netzwerk 1 bis Netzwerk *n* der Reihe nach innerhalb jedes Segments
- ☐ Element auf Element innerhalb eines Netzwerks, vom obersten, linken Element des Kontaktplans von oben nach unten gehend, dann weiter von rechts nach links

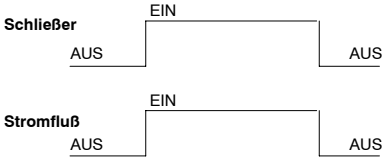
Schaltlogikelemente

In der Kontaktplan-Programmierung werden drei allgemeine Typen von Schaltlogikelementen benutzt — Kontakte, Spulen und Verbindungen. Jedes Schaltlogikelement belegt ein Element im Kontaktplannetz.

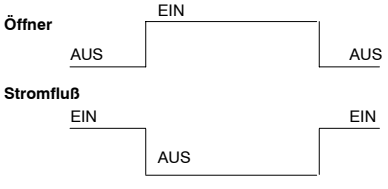
Schaltkontakte

Kontakte werden dazu benutzt, den Stromfluß in einem Kontaktplanlogik-Programm zu ermöglichen oder zu unterbinden. Vier Arten von Kontakten können eingesetzt werden:

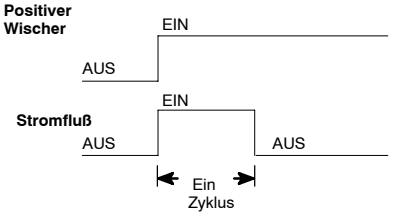
- ☐ Der Schließer, der Stromfluß ermöglicht, wenn seine Referenzspule oder sein Referenzeingang EIN ist:



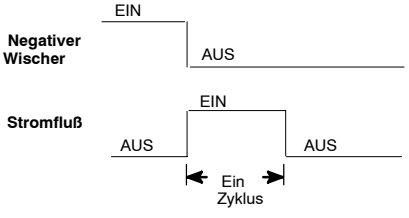
- ☐ Der Öffner, der Stromfluß ermöglicht, wenn seine Referenzspule oder sein Referenzeingang AUS ist:



- ☐ Der positive Wischer, der Stromfluß nur für einen Zyklus ermöglicht, in dem Moment, in dem sich der Zustand des Kontakts oder der Spule von AUS nach EIN ändert:



- ☐ Der negative Wischer, der Stromfluß nur für einen Zyklus ermöglicht, in dem Moment, in dem sich der Zustand des Kontakts oder der Spule von EIN nach AUS ändert:



Die untenstehende Tabelle zeigt die Symbole, die in der Kontaktplanlogik für die Darstellung von verschiedenen Kontakten benutzt werden.

Element	Symbol	Funktion	Speicherbelegung
Schließer		Stromfluß, wenn entspr. Referenzspule oder -Eingang EIN ist	Kann als Referenz eine logische Spule in einem 0*x* Register haben oder einen diskreten Eingang in einem 1*x* Register
Öffner		Stromfluß, wenn entspr. Referenzspule oder -Eingang AUS ist	Kann als Referenz eine logische Spule in einem 0*x* Register haben oder einen diskreten Eingang in einem 1*x* Register
Positiver Wischer		Stromfluß für einen Zyklus, wenn Kontakt oderSpule von AUS nach EINwechselt	Kann als Referenz eine logische Spule in einem 0*x* Register haben oder einen diskreten Eingang in einem 1*x* Register
Negativer Wischer		Stromfluß für einen Zyklus, wenn Kontakt oderSpule von EIN nach AUSwechselt	Kann als Referenz eine logische Spule in einem 0*x* Register haben oder einen diskreten Eingang in einem 1*x* Register

Normale und speicherresidente Spulen

Element	Symbol	Funktion	Speicherbelegung
Normale Spule		schaltet AUS sobald Strom ausgeschaltet wird	Ein diskreter Ausgangswert, repräsentiert durch eine 0*x* Referenzzahl. Kann intern im Logikprogramm oder extern für einen diskreten Ausgang benutzt werden
Speicherresidente Spule		Spule nimmt in den gleichen Zustand wieder ein, sobald Strom wieder eingeschaltet wird	Ein diskreter Ausgangswert, repräsentiert durch eine0*x* Referenzzahl. Kann intern im Logikprogramm oder extern für einen diskreten Ausgang benutzt werden

Eine Spule ist ein diskreter Ausgangswert, der durch ein 0*x* Referenzbit dargestellt wird. Da Ausgangswerte im Status-RAM der CPU aktualisiert werden, kann eine Spule intern in einem Logikprogramm benutzt werden oder, mittels des E/A-Verzeichnisses, extern für eine diskrete Ausgangseinheit des Steuerungssystems.

Abhängig vom Stromfluß kann der Zustand einer Spule entweder EIN oder AUS sein. Wenn die Spule den Zustand EIN hat, läßt sie Strom zu einem diskreten Ausgangsschaltkreis fließen, oder sie ändert den Zustand eines internen Schaltkontakts im Status-RAM.

Es gibt zwei Arten von Spulen — *normale* Spulen und *speicherresidente* Spulen. Wenn bei einer normalen Spule Strom eingeschaltet oder nach einem Ausschaltvorgang erneut eingeschaltet wird, wird jeder vorher in der Spule gespeicherte Wert vor dem ersten Logikzyklus der SPS gelöscht. Bei speicherresidenten Spulen bleibt hingegen ein vorher in der Spule gespeicherter Wert erhalten.

Darstellung von Spulen in einem Netzwerk

Ein Kontaktplan-Netzwerk kann maximal sieben Spulen enthalten. In der elften Spalte sind nur Spulen und keine anderen Logikelemente zulässig. Wenn eine Spule in einer anderen als der elften Spalte erscheint, dürfen in den Gittern rechts von der Spule keine anderen Logikelemente erscheinen.

Vertikale und Horizontale Verbindungen

Verbindungen sind einfache, geradlinige Verbindungen zwischen Befehlsblöcken und/oder Kontakten eines Kontaktplan-Netzwerks.

Eine *vertikale* Verbindung verbindet Kontakte oder Befehlsblöcke, die in einer Netzwerkspalte übereinander liegen. Vertikale Verbindungen können auch dazu benutzt werden, Eingänge oder Ausgänge zu verbinden, um Entweder/Oder-Bedingungen zu erstellen, wie die unten dargestellte. Wenn zwei Kontakte durch eine vertikale Verbindung verbunden sind, fließt Strom, sobald einer der Kontakte oder beide Strom erhalten. Eine vertikale Verbindung belegt keinen Platz im Anwenderspeicher.

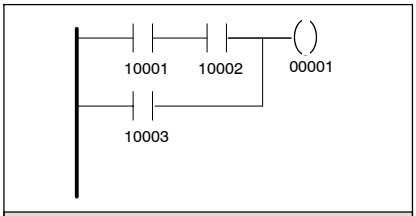
Horizontale Verbindungen werden dazu benutzt, ein Gitter im Kontaktplannetzwerk zu erweitern, ohne den Stromfluß zu unterbrechen. Jede horizontale Verbindung, die in einem Programm benutzt wird, belegt ein Wort im Anwender-Logikspeicher.

Auf der folgenden Seite sind zwei Beispiele abgebildet, die den gemeinsamen Gebrauch von horizontalen und vertikalen Verbindungen zusammen mit Schaltkontakten für die Erstellung eines Logikprogrammes zeigen.

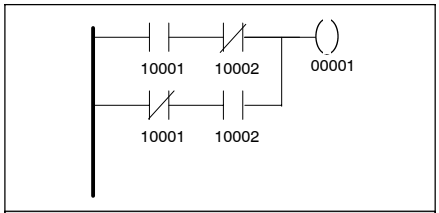
Das erste Beispiel ist eine einfache Entweder/Oder-Bedingung — das oberste Gitter des Kontaktplans enthält zwei Schließer (10001 und 10002), das untere Gitter enthält einen einzelnen Kontakt (10003), an den sich eine horizontale Verbindung anschließt. Am Ende der zweiten Spalte verbindet eine vertikale Verbindung die zwei Gitter. Der Strom kann durch das Netzwerk fließen und Spule 00001 mit Strom versorgen, sobald*entweder* die Kontakte 10001 und 10002 Stromfluß haben *oder* oder Kontakt 10003 Stromfluß hat.

Das zweite Beispiel zeigt eine ausschließende ODER-Schaltung mit ähnlichen Kontakten und Verbindungen. Diese Schaltung kann dazu benutzt werden, die Stromversorgung von Spule 00001 zu verhindern, wenn zwei Bedingungen, dargestellt durch Kontakt 10001 und Kontakt 10002, gleichzeitig aktiviert werden.

Bei beiden Beispielen werden die vertikalen Verbindungen, die keinen Platz im Anwender-Programmspeicher belegen, als Teil des Elements behandelt, in dem Kontakt 10002 programmiert wurde.

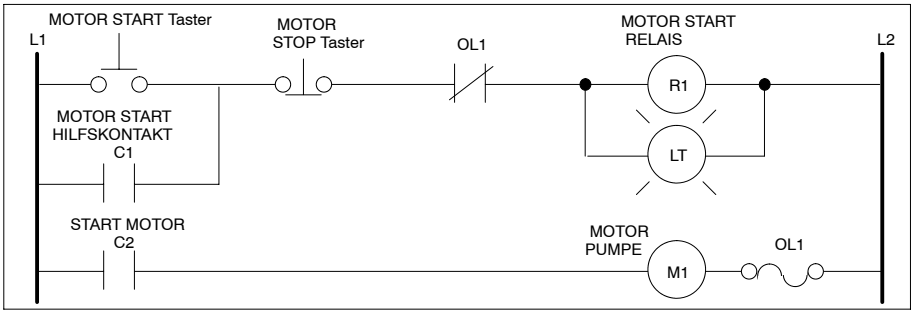


Beispiel 1: Entweder/Oder Schaltlogik



Beispiel 2: Ausschließende ODER Schaltlogik

Anwendungsbeispiel: Start/Stop-Schaltung für einen Motor



Obenstehend sehen Sie ein Beispiel für eine Standarddarstellung einer verdrahteten elektrischen Schaltung, bei der ein Motor durch Betätigung von Start/Stop-Drucktasten aktiviert wird.

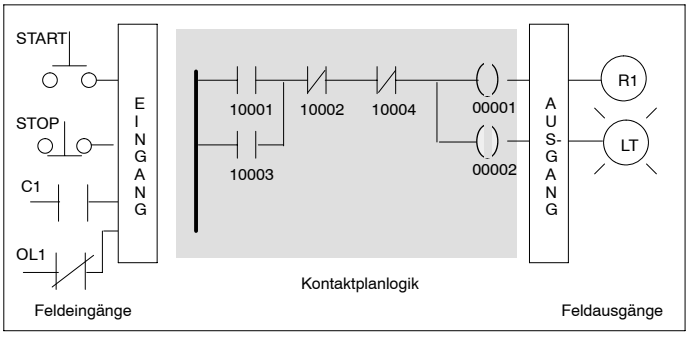
Das Betätigen der Drucktaste *Motor Start* aktiviert die Stromversorgung des Motor-Steuerrelais R1 und schließt Kontakt C2, um den Motor M1 zu starten. Die Hilfskontakte des Motor-Steuerrelais C1 werden ebenfalls geschlossen, wodurch die Motor Start/Stop-Schaltung im Zustand EIN einrastet. Zwei Bedingungen können einen Abfall des Relais C1 verursachen:

- □ Eine Überlastung (OL1) des Motors M1
- □ Die Taste *Motor Stop* wird betätigt.

Betrachten wir jetzt die Umsetzung der gleichen Schaltung in ein Kontaktplan-Netzwerk bestehend aus Kontakten, Spulen und Verbindungen.

In dem Bild unten ist zu erkennen, daß die Reihenfolge der Arbeitsweise bei dem Entwurf der Motor Start/Stop-Schaltung für eine SPS im wesentlichen gleich bleibt. Der größte Unterschied ist der, daß alle E/A-Punkte direkt mit Eingangs-/Ausgangseinheiten des SPS-Systems verdrahtet werden und die eigentliche Steuerung in der SPS in Kontaktplanlogik programmiert wird.

Die Implementierung der Kontaktplanlogik ermöglicht größere Flexibilität bei der Steuerung und verringert die Entwicklungszeit, da die gesamte *Kabelverdrahtung* zwischen den zu steuernden Punkten elektronisch ausgeführt wird.



Kapitel 4
Zähler und Timer

- ☐ Zählerbefehle
- ☐ Timerbefehle
- ☐ Anwendungsbeispiel: Eine Echtzeit-Uhr mit einem Millisekunden-Timer

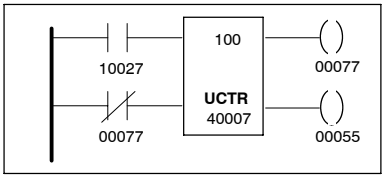
Zählerbefehle

Es gibt zwei Zählerbefehle. Der Vorwärtszähler (UCTR) zählt von 0 bis zu einem vorgegeben Wert, und der Abwärtszähler (DCTR) zählt von einem vorgegebenen Wert abwärts bis 0. Beides sind Zweierelement-Befehle.

Befehl	Struktur	Eingänge (E)	element	Ausgänge (A)	Funktion
Vorwärtszähler	E — 3*x*, 4*x*, oder K* — A E — **UCTR** 4*x* — A	*Oben:* EIN initiiert Zähler *Unten:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Zähler Vorgabe *Unten:* Zählerstand	*Oben:* Zählerstand = Vorgabe *Unten:* Zählerstand < Vorgabe	Zählt aufwärts von 0 bis zu einem vorgegebenen Wert
Abwärtszähler	E — 3*x*, 4*x*, oder K* — A E — **DCTR** 4*x* — A	*Oben:* EIN initiiert Zähler *Unten:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Zähler Vorgabe *Unten:* Zählerstand	*Oben:* Zählerstand = 0 *Unten:* Zählerstand > Vorgabe	Zählt abwärts von einem vorgegebenen Wert bis 0
*K ist eine Integer-Konstante im Bereich 1 ... 999.					

Beispiel für einen einfachen Vorwärtzähler

Wenn Kontakt 10027 Strom erhält, wird die Stromversorgung des oberen Eingangs in den UTCR aktiviert. Da Kontakt 00077 ebenfalls Strom erhält, wird der Befehl aktiviert. Jedes Mal, wenn der Kontakt 10027 vom Zustand AUS zu EIN wechselt, erhöht sich der Zählerstand um den Wert 1. Wenn der Wert 100 erreicht wird, schaltet der obere Ausgang den Strom durch; Spule 00077 erhält Strom und Spule 00055 wird stromlos. Wenn Spule 00077 stromführend wird, erhält Kontakt 00077 keinen Strom mehr und der Zählerstand wird beim nächsten Zyklus auf Null zurückgesetzt. Beim nächsten Zyklus wird Spule 00077 stromlos, Kontakt 00077 wird stromführend und der UTCR dadurch aktiviert.



Timerbefehle

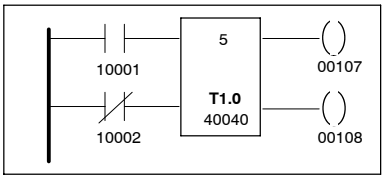
Die vier Timerbefehle können dazu benutzt werden, Ereignisse zeitlich zu steuern oder in eine Anwendung Verzögerungen einzubauen. Die ersten drei sind Zweierelement-Befehle, der Millisekunden-Timer ist ein Dreierelement-Befehl.

Befehl	Struktur	Eingänge (E)	element	Ausgänge (A)	Funktion
Timer für Sekunden	E — 3*x*, 4*x*, oder K* — A E — **T1.0** 4*x* — A	*Oben:* EIN, wenn Eingang unten = 1 *Unten:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Timer Vorgabe *Unten:* Zählerzeit	*Oben:* Zeit = Vorgabe *Unten:* Zeit < Vorgabe	Timer inkrementiert in Intervallen von einer Sekunde
Timer für Zehntelsekunden	E — 3*x*, 4*x*, oder K* — A E — **T0.1** 4*x* — A	*Oben:* EIN, wenEingang unten = 1 *Unten:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Timer Vorgabe *Unten:* Zählerzeit	*Oben:* Zeit = Vorgabe *Unten:* Zeit < Vorgabe	Timer inkrementiert in Intervallen von 0.1 s
Timer für Hundertstelsekunden	E — 3*x*, 4*x*, oder K* — A E — **T.01** 4*x* — A	*Oben:* EIN, wennEingang unten = 1 *Unten:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Timer Vorgabe *Unten:* Zählerzeit	*Oben:* Zeit = Vorgabe *Unten:* Zeit < Vorgabe	Timer inkrementiert in Intervallen von 0.01 s
Timer für Millisekunden	E — 3*x*, 4*x*, oder K* — A E — 4*x* — A **T1MS** 0001	*Oben:* EIN, wenn mittlerer Eingang = 1 *Mitte:* 0 = Rücksetzen 1 = Freigabe	*Oben:* Timer Vorgabe *Mitte:* Zählerzeit *Unten:* Immer auf einen konstanten Wert von 1 eingestellt	*Oben:* Zeit = Vorgabe *Mitte:* Zeit < Vorgabe	Timer inkrementiert in Intervallen von 1 ms

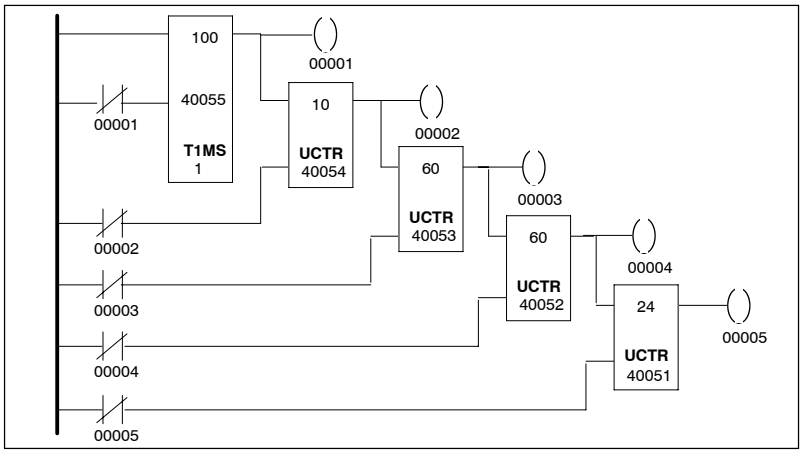
*K ist eine Integer-Konstante im Bereich 1 ... 999.

Beispiel eines Timers für eine Sekunde

Hier ist Kontakt 10002 geschlossen, d.h. der Timer wird freigegeben und der Wert im Register 40040 ist 0. Spule 00108 ist EIN und Spule 00107 AUS. Wenn Kontakt 10001 geschlossen wird, erhöht sich der Zählerstand in Register 40040 in Ein-Sekunden-Intervallen bis der Wert 5 erreicht ist. Spule 00107 schaltet auf EIN und Spule 00108 AUS. Wird Kontakt 10002 geöffnet, wird der Wert des Registers 40040 auf 0, Spule 00107 auf AUS und Spule 00108 auf EIN gesetzt.



Anwendungsbeispiel: Eine Echtzeit-Uhr mit einem Millisekunden-Timer



Dieses Beispiel zeigt die Kontaktplanlogik für eine Echtzeit-Uhr mit einer Genauigkeit im Bereich von Millisekunden. Der Befehl T1MS wird so programmiert, daß in Intervallen vom 100 ms Stromfluß hergestellt wird. Darauf folgt eine Reihenschaltung von vier Vorwärtszählern, die die Zeit in den Einheiten Hunderstel-, Zehntel-Sekunden, Sekunden, Minuten und Stunden speichern.

Wenn die Abarbeitung der Logik beginnt, wird in Register 40055 des T1MS-Blocks der aktuelle Zeitwert erhöht. Nach zehn Inkrementierungen von je einer Millisekunde wird der obere Ausgang stromführend und aktiviert Spule 00001.

An diesem Punkt wird Register 40053 des Timers auf den Wert 0 zurückgesetzt. Der bisherige Zählerstand in Register 40054 des ersten UCTR-Blocks wird um 1 erhöht, was bedeutet, daß 10 ms vergangen sind.

Da nun der Zählerwert von T1MS nicht mehr der Timer-Vorgabe entspricht, beginnt der Timer die Zeit in ms wieder neu zu zählen.

Wenn der Zählerstand in Register 40054 des ersten UCTR-Befehls den Wert 10 erreicht, wird der obere Ausgang dieses Befehlsblocks stromführend und aktiviert Spule 00002. Der Wert des Registers 400054 wird anschließend auf 0 zurückgestellt und der Zählerstand in Register 40051 des zweiten UCTR-Blocks um 1 erhöht.

Durch die Erhöhung des Zählerstandes in jedem Zähler kann die Tageszeit in den folgenden fünf Halteregistern ausgelesen werden:

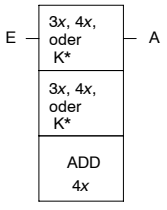
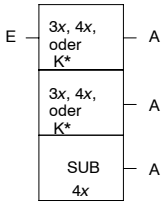
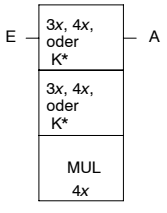
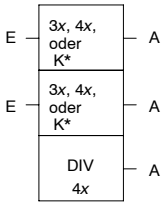
Register	Zeiteinheit
40055	Hunderstelsekunden (0 ... 10)
40054	Zehntelsekunden (0 ... 10)
40053	Sekunden (0 ... 60)
40052	Minuten (0 ... 60)
40051	Stunden (0 ... 24)

Kapitel 5
Grundlegende Mathematikbefehle

- ☐Integer-Mathematikbefehle
- ☐Anwendungsbeispiel: Umwandlung von Fahrenheit in Celsius

Integer-Mathematikbefehle

Für mathematische Rechenoperationen mit ganzen Zahlen werden Standardbefehle für Addition, Subtraktion, Multiplikation und Division bereitgestellt. Jeder der vier Befehle ist ein Dreierelement-Befehl.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Integer Addition		*Oben:* EIN führt zu einer (Wert 1) + (Wert 2) Operation	*Oben:* Wert 1 Mitte: Wert 2 Unten: Summe	Oben: Summe> 9999	Addiert die Werte des oberenund mittleren Elements und speichert das Ergebnis in einem 4*x* Register des unteren Elements
Absolute Integer Subtraktion(ohneVorzeichen)		*Oben:* Ein führt zu einer (Wert 1) - (Wert 2) Operation	*Oben:* Wert 1 Mitte: Wert 2 Unten: Differenz	*Oben:* Wert 1 > Wert 2 Mitte: Wert 1 = Wert 2 Unten: Wert 1 < Wert 2	Subtrahiert den Wertdes mittleren Elements vondem des oberen und speichert die Differenzin einem 4*x* Registerdes unteren Elements
Integer Multiplikation		*Oben:* EIN führt zu einer (Wert 1) x (Wert 2) Operation	*Oben:* Wert 1 Mitte: Wert 2 Unten: Produkt (höher- wertige Ziffern)	*Oben:* Echo der oberen Eingabe	Multipliziert die Werte des obersten und mittlerenElements und speichert das Produkt in zwei aufeinander-folgenden 4*x* Registern
Integer Division mit Rest		*Oben:* EIN führt zu einer (Wert 1) / (Wert 2) Operation Mitte: 0 = Bruch- rest 1 = Dezimal- rest	*Oben:* Wert 1** Mitte: Wert 2 Unten: Ergebnis (Rest in Reg 4x + 1)	*Oben:* Division erfol- greich Mitte: Wenn Ergebnis > 9999 wird ein Wert von 0 zurückgegeben Unten: Wert 2 = 0	Dividiert den Wert des oberen Elementsdurch den Wert des mittleren Elementsund speichert das Ergebnis im 4*x* Register des unteren Elements und den Rest in Register 4*x* + 1

*K ist eine Integer-Konstante im Bereich 1 ... 999.

** Wenn Wert 1 des DIV Befehls in den 3*x* oder 4*x* Registern gespeichert wird, ist das Register im oberen Element das erste zweier aufeinanderfolgender Register. Die höherwertige Hälfte von Wert 1 wird in dem angezeigten Register gespeichert (3*x* oder 4*x*) und die niederwertige Hälfte von Wert 1 wird in dem nächsten folgenden Register (3*x* + 1 oder 4*x* + 1)gespeichert.

Die Blöcke MUL und DIV benötigen zwei aufeinanderfolgende Register im unteren Element. Das erste der beiden Register erscheint im Block, das Vorhandensein des zweiten Registers wird impliziert.

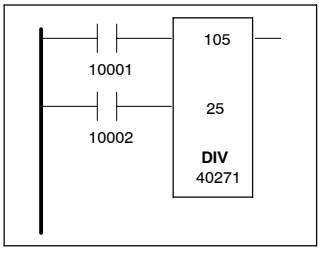
Im MUL Befehlsblock wird der höherwertige Anteil des berechneten Produkts im ersten unteren Element des Registers gespeichert, der niederwertigere Anteil des Produktes im zweiten unteren Element des Registers.

Beim DIV Befehlsblock wird der Quotient im ersten unteren Element des Registers und der Rest im zweiten unteren Element des Registers gespeichert. Wenn Sie in einem DIV-Befehl nicht eine Konstante als Wert des oberen Elements benutzen, dann muß der Wert in zwei aufeinanderfolgenden 3*x* oder 4*x* Registern abgelegt werden. Die höherwertige Hälfte des Wertes wird im angezeigten Register, die niederwertige Hälfte des Wertes wird im implizierten Register gespeichert.

Wenn z.B. der Wert des obersten Elements 105 ist und statt als Konstante in zwei aufeinanderfolgenden Registern, 40025 und 40026 plaziert werden soll, würde Register 40025 alle Nullen enthalten und Register 40026 den Wert 105.

Ein DIV Beispiel

Es folgt ein Beispiel für eine DIV Operation, bei der der obere Elementwert, 105, durch den Wert des mittleren Elements, 25, geteilt wird. Der Quotient (4) wird in Register 4027, der Rest (5) in Register 40272 gespeichert.



Wenn der mittlere Eingang — Kontakt 10002 — geöffnet ist, wird der Rest als Bruchteil ausgedrückt (0005); ist Kontakt 10002 geschlossen, wird der Rest als Dezimalzahl dargestellt (2000).

Anwendungsbeispiel: Umwandlung von Fahrenheit in Celsius

Dieses Beispiel implementiert die Formel

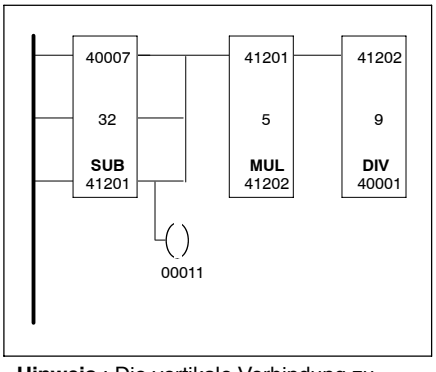
$$°C = (°F - 32) \times {}^{5}/_{9}$$

Wenn der obere Eingang des SUB Befehlsblocks Strom erhält, wird der Wert des mittleren Elements, 32, von dem Wert in Register 30001 subtrahiert, das eine Zahlenangabe in Grad Fahrenheit enthält. Die Differenz wird in Register 41201 abgelegt.

Daraufhin erhält der obere Eingang des MUL Befehlsblocks Strom, unabhängig davon, ob das Ergebnis der Subtraktion positiv, negativ oder 0 ist. Wenn das Subtraktionsergebnis negativ ist, wird Spule 00011 aktiviert und zeigt einen negativen Wert an.

Der Wert im obersten Element des MUL-Blocks — Register 41201 — wird daraufhin mit 5 multipliziert und das Produkt in Register 41202 abgelegt und implizit in Register 41203.

Anschließend erhält das oberste Element des DIV-Befehlsblocks Strom und der Wert in den Registern 41202 und 41203 wird durch 9 geteilt. Der Quotient, die in Grad Celsius umgewandelte Temperaturangabe, wird in Register 4001 gespeichert (und der Rest im impliziten Register 40002).



Hinweis : Die vertikale Verbindung zu Spule 00011 (Anzeige eines negativen Wertes) muß links von den vertikalen Verbindungen angeordnet werden, die die drei SUB-Block Ausgänge verbinden.

Kapitel 6
Datenmanagement-Befehle

- ☐Verschieben von Register- und Tabellendaten
- ☐Anlage eines FIFO Stack
- ☐Suche in einer Tabelle
- ☐Verschieben eines Datenblocks

Verschieben von Register- und Tabellendaten

Es gibt drei Standard-Befehlsblöcke für das Verschieben von Daten, die in Registern und Tabellen gespeichert sind:

- ☐ DX-Verschiebung Register-zu-Tabelle (R→T)
- ☐ DX-Verschiebung Tabelle-zu-Register (T→R)
- ☐ DX-Verschiebung Tabelle-zu-Tabelle (T→T)

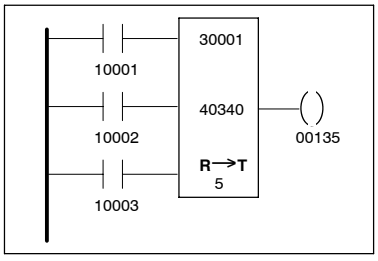
Ein Modicon Micro SPS-System kann den Transfer eines Registers pro Zyklus für jeden Befehl in einem Kontaktplanlogikprogramm ausführen.

Es handelt sich jeweils um einen Dreierelement-Befehl.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Register-zu-TabelleVerschiebung	E — 0*x*, 1*x*, * 3*x* oder 4*x* — A E — *4x* — A E — R→T K**	*Oben:* EIN verschiebt Daten und inkrementiert den Zeiger Mittel: EIN blockiert den Zeiger Unten: EIN setzt den Zeiger auf 0 zurück	*Oben:* Quellregister Mittel: Zeiger auf das Zielregister (4x + 1) in der Zieltabelle Unten: Tabellenlänge*	Oben: Kopie des oberen Eingangs Mittel: Zeiger = Tabellenlänge	Kopiert ein 16-Bit Musterin einem Quellregister zu einem Register in der Zieltabelle; auf das Zielregister wird durch das 4*x* Register des mittleren Elementsverwiesen
Tabelle-zu-Register Verschiebung	E — 0*x*, 1*x*, * 3*x* oder 4*x* — A E — *4x* — A E — T→R K**	*Oben:* EIN verschiebt Daten und inkrementiert den Zeiger Mittel: EIN blockiert den Zeiger Unten: EIN setzt den Zeiger auf 0 zurück	*Oben:* Quelltabelle Mittel: Zeiger auf das Zielregister (4x + 1) Unten: Tabellenlänge*	Oben: Kopie des oberen Eingangs Mittel: Zeiger = Tabellenlänge	Kopiert das Bitmuster eines Registers in der Quelltabelle zu einem Zielregister (Register 4*x* + 1 in dem mittleren Element)
Tabelle-zu-Tabelle Verschiebung	E — 0*x*, 1*x*, * 3*x* oder 4*x* — A E — *4x* — A E — T→T K**	*Oben:* EIN verschiebt Daten und inkrementiert Zeiger Mittel: EIN blockiert den Zeiger Unten: EIN setzt den Zeiger auf 0 zurück	*Oben:* Quelltabelle Mittel: Zeiger auf das Zielregister (4x + 1) in der Zieltabelle Unten: Tabellenlänge*	Oben: Kopie des oberen Eingangs Mittel: Zeiger = Tabellenlänge	Kopiert das Bitmuster eines Registers der Quelltabelle in ein Register in der gleichen Position in einer Ziel-Tabelle; auf das Zielregister wird durch das 4*x* Register des mittlern Elements verwiesen

* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß diese ein Mehrfaches von 16 + 1 sein (1, 17, 33, etc.), wodurch der Gebrauch von 16 Bit impliziert wird (1 ... 16, 17 ... 32, 33 ... 48, etc.).

** K ist eine Integer-Konstante im Bereich 1 ... 255.



Das oben gezeigte Kontaktplanlogik-Beispiel verschiebt den in Register 30001 gespeicherten Wert zu einer Zieltabelle bestehend aus fünf Halteregistern, 40341 ... 40345. Bei jedem Zyklus wird ein Wert des 30001 Registers in einen Wert des Tabellenregisters verschoben.

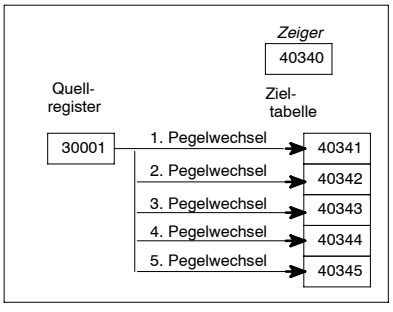
Der auf die Zieltabelle verweisende Zeiger — Register 40340 — ist im mittleren Element des Register-zu-Tabelle Befehlsblocks spezifiziert, die Anzahl der Halteregister der Tabelle, 5, ist im unteren Element festgelegt.

Beim ersten Pegelwechsel des Kontakts 10002 zu EIN werden die aktuellen Inhalte des Registers 30001 in Register 40341 kopiert, in das erste von fünf aufeinanderfolgenden Registern der Zieltabelle. Die erste Tabelle des Zielregisters ist immer das nächste folgende Register, nach der Zeiger-Referenznummer, die im mittleren Element des Befehlsblocks angegeben ist. Bei der Ausführung dieser DX-Verschiebung wird der Wert des Zeigers von 0 auf 1 erhöht.

Beim nächsten Zyklus des Kontakts 10001 werden die Inhalte des Registers 30001 in Register 40432 kopiert, in das zweite Register der Zieltabelle. Der Wert des Zeigerregisters wird von 1 auf 2 erhöht.

Dieser Vorgang wird weiter ausgeführt, bis der Inhalt des Registers 30001 in das Register 40435 der Tabelle kopiert ist und der Zeigerwert 5 beträgt. An dieser Stelle wird der mittlere Ausgang des Blocks stromführend und aktiviert Spule 00135.

Solange der Wert des Zeigers der im unteren Element des Blocks angegebenen Tabellenlänge entspricht, sind keine weiteren Register-zu-Tabelle Verschiebungen möglich.



Falls, nach dem zweiten Pegelwechsel von Kontakt 10001, Kontakt 10002 stromführend werden soll, müßte der Zeigerwert eingefroren werden, d.h. er kann dann nicht erhöht oder erniedrigt werden. Anschließende Pegelwechsel des Kontakts 10002 würden dazu führen, daß der aktuelle Wert des Registers 30001 in Register 40343 kopiert würde.

Wenn Kontakt 10003 stromführend wird, wird der Wert des Zeigers auf 0 zurückgestellt.

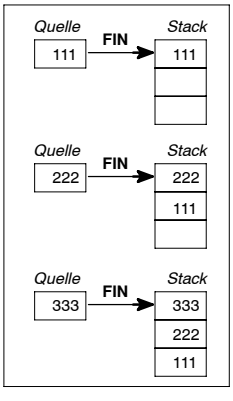
Anlage eines FIFO Stack

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
First In in Warte-Stack	E — 0*x*, 1*x*, * 3*x* oder 4*x* — A 4*x* — A FIN K** — A	*Oben:* EIN fügt ein Bitmuster oben in den Stack ein	Oben: Das Quellregister im Stack Mittel: Zeiger auf das Register im Stack, wo die Quellbits eingefügt werden Unten: Stacklänge*	Oben: Kopie des oberen Eingangs Mittel: Stack ist voll Unten: Stack ist leer	Kopiert ein 16-Bit Muster in ein Register oben im Stack; die Tabelle beginnt beiRegister 4*x* + 1 des mittleren Elements
First Out aus einem Warte-Stack	E — 4*x* — A 0*x* oder 4*x* — A FOUT K** — A	*Oben:* EIN entfernt das Bitmuster unten aus dem Stack	Oben: Zeiger auf das Quellregister im Stack Mittel: Zielregister, zu dem Quellbits verschoben werden Unten: Stacklänge*	Oben: Kopie des oberen Eingangs Mittel: Stack ist voll Unten: Stack ist leer	Verschiebt das Bitmuster im unteren Register des Stack zu einem Zielregister aus dem Stack heraus

** K ist eine Integer-Konstante im Bereich 1 ... 255.

* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß es ein Mehrfaches von 16 + 1 sein(1, 17, 33, etc.) und impliziert die Benutzung von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

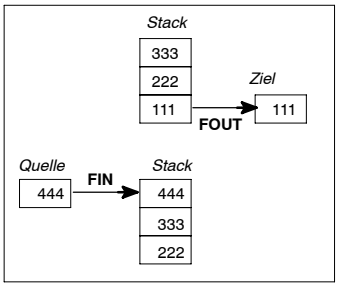
Die beiden oben dargestellten Befehle ermöglichen Ihnen, Daten in einem Stack in einer Warteschlange nach dem Prinzip First In (Erster rein)/First Out (Erster raus) einzuzuordnen. Der FIN-Befehl kopiert das Bitmuster eines Registers oder von 16 diskreten in ein Register an der Spitze der Tabelle (oder des Stack) der Halteregister.



Der Befehl FOUT verschiebt die Bitmuster im Stack nach unten, dann aus dem Stack hinaus in eine Zieltabelle.

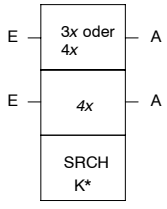
Warnung: FOUT übergeht alle auf disabled gesetzten Spulen einer Zieltabelle, ohne sie auf enabled zu ändern. Wenn eine Spule aus Reparatur- oder Wartungsgründen auf disabled gesetzt wurde, besteht Verletzungsgefahr, da der Zustand dieser Spule sich durch die FOUT-Operation verändern kann.

Wenn Sie einen FIFO-Stack in der Kontaktplanlogik ausführen, sollte der FOUT-Befehl in jedem Zyklus vor der FIN-Anweisung ausgeführt werden, so daß die ältesten Daten des Stacks in die Zieltabelle geschrieben werden, bevor die neuesten Daten in die Stack-Warteschlange eingereiht werde. Wird der FIN-Block als erstes ausgeführt, wird der Versuch ignoriert, Daten in einen gefüllten Stack einzugeben.



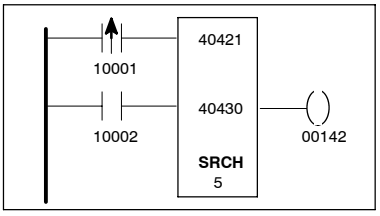
Suche in einer Tabelle

Der SRCH-Befehl ermöglicht Ihnen, in einer Registertabelle nach einem bestimmten Bitmuster zu suchen, das in einem der Tabellenregister enthalten ist. SRCH ist ein Dreierelement-Befehl.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Tabellensuche		*Oben:* EIN initiiert den Suchvorgang Mittel: 0 = Suche von Anfang an 1 = Suche von der letzten Fundstelle an	*Oben:* Erstes Register in der Quelltabelle Mittel: 4x Zeiger auf den Ort in der Tabelle des Registers,das den zu suchenden Wert enthält; das nächste Register, 4x + 1, enthält den zu suchenden Wert Unten: Tabellenlänge*	Oben: Kopie des oberen Eingangs Mittel: Suchbegriff gefunden	Sucht in einer Tabelle von Registern nach einem Bitmuster, das im direkt auf den Zeiger im mittleren Element folgenden Register spezifiziert ist

* K ist eine Integer-Konstante im Bereich 1 ... 255.

Beispiel für eine SRCH Operation



Die Quelltabelle, in der gesucht werden soll, ist fünf Register lang und beginnt bei Halteregister 40421. Der Inhalt des Tabellenregisters sieht so aus:

Register der Quelltabelle		Register-inhalt
40421	=	1111
40422	=	2222
40423	=	3333
40424	=	4444
40425	=	5555

Das zu suchende Bitmuster ist 3333, das ist der Wert, der in Register 40431 eingegeben ist (das Register direkt nach dem Zeigerregister im mittleren Element).

Wenn der Zustand des Kontakts 10001 von AUS nach EIN wechselt, sucht die Logik in der Quelltabelle nach dem Register mit dem Inhalt 3333. Ist der Wert gefunden worden (in Register 40423), wird der Zeigerwert in Register 40430 auf 3 gesetzt, wodurch angezeigt wird, daß das dritte Register in der Quelltabelle den gesuchten Wert enthält. Für die Dauer eines Zyklus wird außerdem Spule 00142 aktiviert.

Verschieben eines Datenblocks

Der Befehl Blockverschiebung (BLKM) kopiert in einem Logikzyklus den gesamten Inhalt einer Quelltabelle von Registern zu einer Zieltabelle. BLKM ist ein Dreierelement-Befehl.

Warnung: BLKM übergeht alle auf disabled gesetzten Spulen einer Zieltabelle, ohne sie auf enabled zu ändern. Wenn eine Spule aus Reparatur- oder Wartungsgründen auf disabled gesetzt wurde, besteht Verletzungsgefahr, da der Zustand dieser Spule sich durch die BLKM-Operation verändern kann.

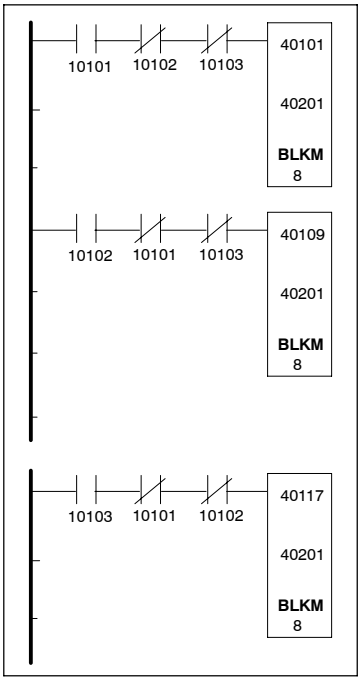
Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Blockverschiebung	E — 0*x*, 1*x*, * 3*x*, oder 4*x* 0*x*** oder 4*x* BLKM K*** — A	*Oben:* EIN initiiert eine Blockverschiebung	*Oben:* Quelltabelle Mittel: Zieltabelle Unten: Tabellenlänge*	Oben: Kopie des oberen Eingangs	Kopiert den gesamten Inhalt einer Tabelle zu einer anderen Tabelle mit Ausgängen oder Halte- registern
*** K ist eine Integer-Konstante im Bereich 1 ... 100.					

Anwendungsbeispiel: Routine zum Laden eines Rezepts mithilfe Blockverschiebung

Ein Kontaktplanlogikprogramm kann eine Sammlung spezifischer Prozessrezepte speichern, von denen jedes in einer einmaligen Speichertabelle aufbewahrt und auf Anforderung in eine Arbeitstabelle geladen wird, von der aus der Auswahlprozess abläuft. Die Rezepte müssen über gleichartige Informationstypen in den entsprechenden Registern verfügen — wenn Information über eine Erhitzungstemperatur im dritten Register des einen Rezepts gespeichert ist, sollten die dritten Register aller anderen Rezepte ebenfalls eine solche Information enthalten.

Bestimmte Rezepte können mit Hilfe des BLKM-Befehls in den Auswahlprozess geladen und auch entfernt werden.

Das im nächsten Bild gezeigte Logikbeispiel enthält eine acht Register umfassende Arbeitstabelle (Register 40201 ... 40208), in der drei verschiedene Rezepte ausführbar sind. Die Auswahl der Rezepte wird über drei Eingangsschalter, die Kontakte 10101, 10102 und 10103, durchgeführt.



Um z.B. Prozess A auszuführen, müssen Sie Schalter 10101 auf EIN schalten und die Kontakte 10102 und 10103 auf AUS. Wenn Eingang 10101 stromführend ist, leitet er den Strom durch die Öffner 10102 und 10103 und als Folge verschiebt der erste BLKM-Block das Rezept für Prozess A von den Registern 40101 ... 40108 zu den Registern 40201 ... 40208.

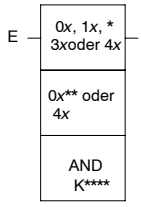
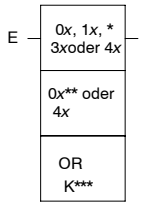
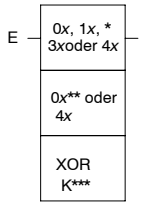
Kapitel 7
Datenmanipulations-Befehle

- ☐Befehle für Boolesche Logik
- ☐Ein Anwendungsbeispiel: einfache Berechnung des Tabellendurchschnitts
- ☐Bit-Komplementbildung in einer Datenmatrix
- ☐Bit-Vergleich in einer Datenmatrix
- ☐Bit-Abfrage und -Veränderung in einer Datenmatrix

Befehle für Boolesche Logik

Es stehen drei Befehle zur Verfügung, die logische Operationen in Form von AND,OR und XR (exklusives OR) durchführen.

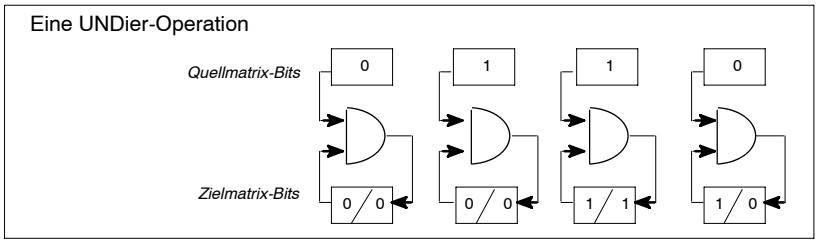
Warnung: Diese booleschen Befehle übergehen alle auf disabled gesetzten Spulen einer Zieltabelle, ohne sie auf enabled zu ändern. Wenn eine Spule aus Reparatur- oder Wartungsgründen auf disabled gesetzt wurde, besteht Verletzungsgefahr, da der Zustand dieser Spule sich durch die logische Operation verändern kann.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Boolsches UND		Oben: Initiiert eine logische UND Operation	Oben: Quellmatrix Mitte: Zielmatrix Unten: Matrixlänge*	Oben: Kopie des oberen Eingangs	UNDiert die Bits in der Quellmatrix mit entsprechend positioniertenBits in der Zielmatrix und plaziert die Ergebnisse in der Zielmatrix, wobei dasursprüngliche Bitmusterüberschrieben wird
Boolsches ODER		Oben: Initiiert eine logische ODER Operation	Oben: Quellmatrix Mitte: Zielmatrix Unten: Matrixlänge*	Oben: Kopie des oberen Eingangs	ODERiert die Bits in der Quellmatrix mit entsprechend positioniertenBits in der Zielmatrix und plaziert die Ergebnisse in der Zielmatrix, wobei das ursprüngliche Bitmuster überschrieben wird
Boolsches XODER		Oben: Initiiert eine logische XODER Operation	Oben: Quellmatrix Mitte: Zielmatrix Unten: Matrixlänge*	Oben: Kopie des oberen Eingangs	XODERiert die Bits in der Quellmatrix mit entsprechend positioniertenBits in der Zielmatrix undplaziert die Ergebnisse in der Zielmatrix, wobei das ursprüngliche Bitmuster überschrieben wird

*** K ist eine Integer-Konstante im Bereich 1 ... 100

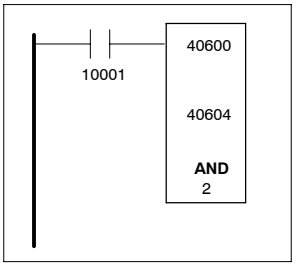
* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß es ein Mehrfaches von 16 + 1 sein (1, 17, 33, etc.) und impliziert den Gebrauch von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

** Wenn 0*x* Referenzen als Ziel benutzt werden, können sie nicht als Spulen programmiert werden, sondern nur als Kontakte, die auf diese Spulennummern verweisen.

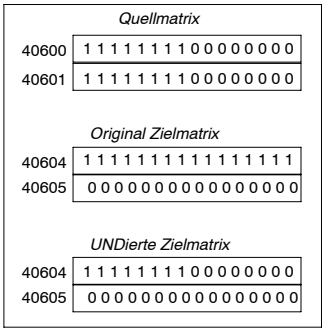


Ein AND-Befehl führt eine logische UNDierung jedes Bits der Quellmatrix mit den entsprechenden Bits der Zielmatrix durch und legt dann die Ergebnisse in der Zielmatrix ab, wobei das vorherige Bitmuster in der Zielmatrix überschrieben wird.

Wenn z. B. Kontakt 10001 in dem unten dargestellten Netzwerk stromführend wird, wird die Bitmatrix, die aus den Registern 40600 und 40601 besteht, mit der Bitmatrix der Register 40604 und 40605 UNDiert.



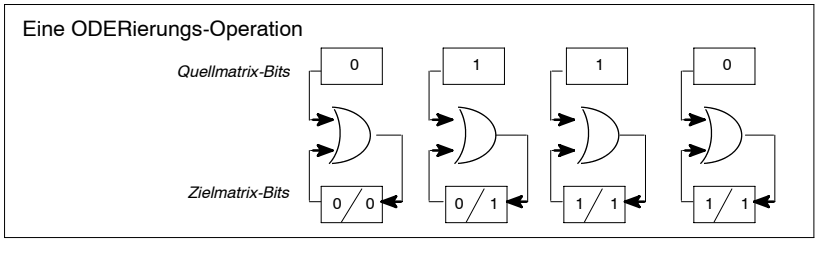
Das Ergebnis wird anschließend in die Register 40604 und 40605 kopiert, wobei das vorher dort vorhandene Bitmuster überschrieben wird.



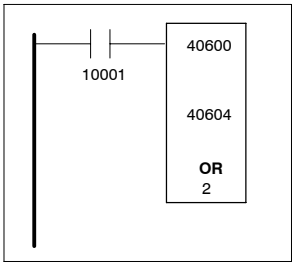
ODER

Auf vergleichbare Weise ODERiert ein OR-Befehl logisch die Bits einer Quellmatrix mit den entsprechenden Bits in der Zielmatrix und überschreibt dann die Zielmatrix mit dem Ergebnis der Operation.

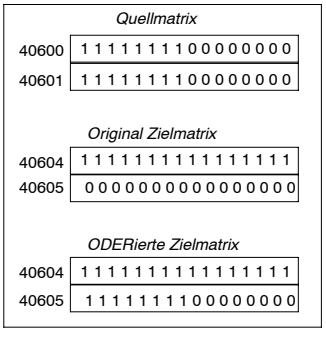
☞ **Hinweis** Ausgänge und Spulen können mit dem OR-Befehl nicht AUSgeschaltet werden.



Wenn die beiden folgenden, oben dargestellten Beispielmatrixen ODERiert werden sollten:



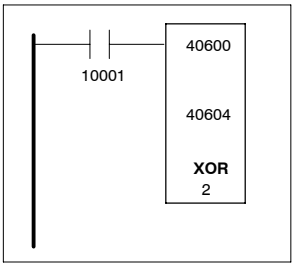
wäre das Ergebnis der Operation:



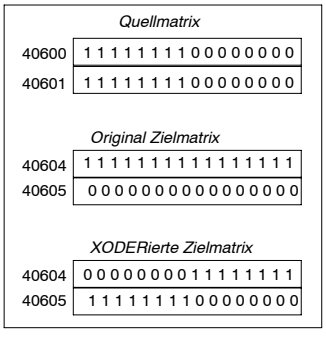
Exklusives ODER

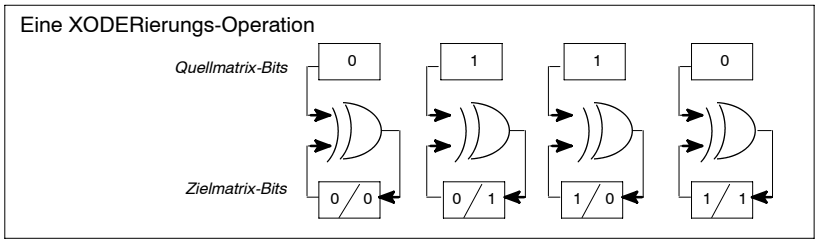
Der XOR-Befehl führt zwischen den Bits der Quellmatrix und den Bits in der Zielmatrix eine XOR-Operation durch und überschreibt dann die Zielmatrix mit dem Ergebnis.

Wenn z.B. mit den gleichen, oben dargestellten Matrixen eine XOR-Operation ausgeführt würde:



wäre das Ergebnis:

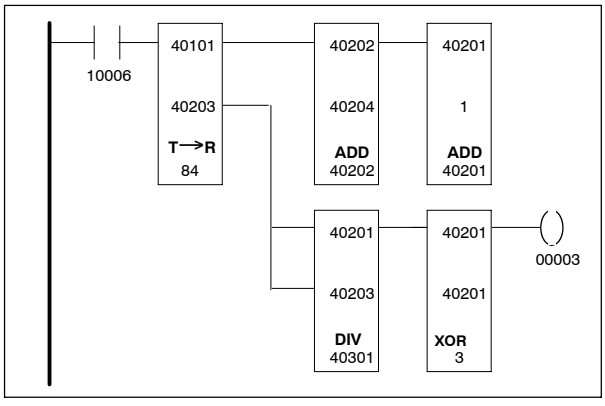




Archivierung der ursprünglichen Zielmatrixwerte

Wenn Sie die ursprünglichen Bitmuster der Zielmatrixregister speichern wollen, kann das mit Hilfe des BLKM-Befehls geschehen, mit dem Sie die Information in eine andere Tabelle kopieren können, bevor Sie die boolsche Logikoperation ausführen.

Ein Anwendungsbeispiel: einfache Berechnung des Tabellendurchschnitts



Es folgt eine Anwendungsroutine, die drei Integer-Berechnungen mit einem Datentransfer und einem XOR-Befehl verbindet. Es wird der Durchschnittswert der 84 Werte berechnet, die in der Tabelle der Register 40101 ... 40184 gespeichert sind.

Wenn Kontakt 10001 stromführend wird, erhält das oberste Element des Tabelle-zu-Register-Befehls Strom und leitet daher den Datentransfer ein. Der Wert im ersten Register der Tabelle wird in das mittlere Element des ersten AND-Befehls kopiert und der Wert des Tabellenzeigers im mittleren Element sowohl in dem Tabelle-zu-Register-Befehl als auch dem DIV-Befehl um 1 erhöht. Da der oberste Ausgang des Tabelle-zu-Register-Befehls stromführend wird, erhält der erste AND-Block Strom und addiert den Wert in Register 40204 zu dem Wert in Register 40202 (welcher anfangs 0 ist). Anschließend wird der vorherige Wert in Register 40202 mit der Summe dieser Addition überschrieben.

Diese Routine geht so weiter, bis alle Werte der Tabelle mit den 84 Registern zusammenaddiert worden sind. An dieser Stelle hat der Zeiger im mittleren Element des Tabelle-zu-Register-Befehls den Wert 84, wodurch der mittlere Ausgang dieses Blocks stromführend wird und den DIV-Befehl aktiviert.

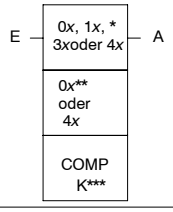
Die Werte in den Registern 40201 (alle 0, die den höherwertigen Anteil der Summe aller Registerwerte der Tabelle darstellen) und 40202 (der niederwertigere Teil der Summe) werden durch 84 geteilt. (Da der mittlere Eingang des DIV-Befehls stromführend ist, wird der Rest als Dezimalzahl ausgedrückt.) Das Ergebnis der DIV-Operation ist der Durchschnittswert der aktuell in allen 84 Registern der Tabelle gespeicherten Werte.
Wenn der oberste Ausgang des DIV-Befehls stromführend wird, wird der XOR-Befehl aktiviert. Dieser führt eine exklusive ODERierung der Werte in den Registern 40201 ... 40203 mit sich selbst durch, löscht die Matrix mit 0-Werten, wodurch angezeigt wird, daß die Operation für die Berechnung des aktuellen Tabellendurchschnitts beendet ist und ein neuer Vorgang beginnen kann.

Bit-Komplementbildung in einer Datenmatrix

Der COMP-Befehl bildet das Komplement des Bitmusters in einer Matrix - d.h. alle 0 werden zu 1 und alle 1 zu 0 verändert-und kopiert das Ergebnis in eine zweite Matrix. Zu einer Matrix kann in einem Zyklusdurchlauf das Komplement gebildet werden.

COMP ist ein Dreierknoten-Befehl.

Warnung: COMP übergeht alle auf disabled gesetzten Spulen einer Zieltabelle, ohne sie auf enabled zu ändern. Wenn eine Spule aus Reparatur- oder Wartungsgründen auf disabled gesetzt wurde, besteht Verletzungsgefahr, da der Zustand dieser Spule sich durch die COMP-Operation verändern kann.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Bit Komplement		Oben: EIN initiiert eine Bit-Komplement Operation	Oben: Quellmatrix Mitte: Zielmatrix Unten: * Matrixlänge	Oben: Kopie des oberen Eingangs	Bildet das Komplement der Bitwerte in der Quellmatrix und plaziert die Ergebnisse in der Zielmatrix

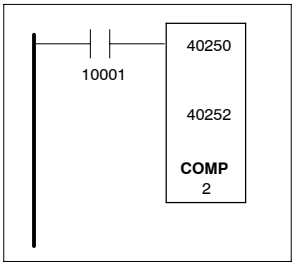
* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß es ein Mehrfaches von 16 + 1 sein (1, 17, 33, etc.) und impliziert den Gebrauch von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

** Wenn 0*x* Referenzen als Ziel benutzt werden, können sie nicht als Spulen programmiert werden, sondern nur als Kontakte, die auf diese Spulennummern verweisen.

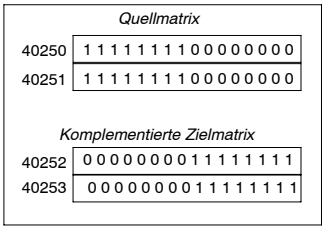
*** K ist eine Integer-Konstante im Bereich 1 ... 100

Beispiel eines Bit-Komplements

Die untenstehende Kontaktplanlogik zeigt einen COMP-Block mit einer Quellmatrix, die aus zwei Registern besteht — 40250 und 40251 — und eine Zielmatrix mit den Registern 40252 und 40253.



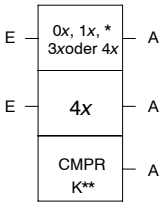
Wenn Kontakt 10001 stromführend wird, führt der Block eine Komplementbildung der Bitwerte des Quellregisters durch und legt die Ergebnisse im Zielregister ab.



Alle Werte, die im Zielregister vor Aufruf des COMP-Befehls gespeichert waren, werden, als Ergebnis der COMP-Operation, mit den komplementierten Werten des Quellregisters überschrieben.

Bit-Vergleich in einer Datenmatrix

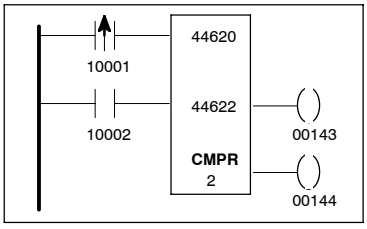
Der CMPR-Befehl vergleicht das Bitmuster in einer Registermatrix mit dem Muster in einer anderen Matrix. Wenn ein Bitwert in einer Matrix nicht mit den an korrespondierender Position in der anderen Matrix befindlichen Bit übereinstimmt, wird im mittleren Element ein Wert mit der Position dieser Matrix abgelegt.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Bit-vergleich		Oben: EIN initiiert einen Bitvergleich Mitte: 0 = Neubeginn bei letztem Vergleichsfehler 1 = Neubeginn am Anfang	Oben: Matrix a Mitte: enthält die Position des aktuellen Vergleichsfehlers des Bits und weist auf Matrix b, die bei 4x + 1 beginnt Unten: * Matrixlänge	Oben: Kopie des oberen Eingangs Mitte: Vergleichsfehler festgestellt Unten: Status des fehlerhaften Bits in Matrix a	Vergleicht Bitmuster in den Matrixen a und b, und gibt Vergleichsfehler an

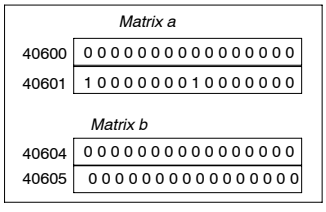
** K ist eine Integer-Konstante im Bereich 1 ... 100

* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß sie ein Mehrfaches von 16 + 1 sein(1, 17, 33, etc.) und impliziert den Gebrauch von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

Beispiel eines Bit-Vergleichs



Dieses Beispiel zeigt einen Bit-Vergleich zwischen zwei Zwei-Register-Matrixen. Matrix a besteht aus den Registern 44620 und 44621, Matrix b umfaßt Register 44623 und 44624:



Matrix a wird mit Matrix b verglichen, bis ein Vergleichsfehler gefunden wird, und zwar Bit für Bit bei jedem Zyklusdurchlauf, in dem Kontakt 10001 von AUS nach EIN wechselt.

Beim ersten Pegelwechsel von Kontakt 10001 werden die Matrixbits bis Bit 17 verglichen, wo der Wert in Matrix a = 1 und der Wert in Matrix b = 0 ist. An dieser Stelle wird der Wert 17 in Register 44622 deponiert, der Vergleich wird angehalten und die Spulen 00143 und 00144 für einen Zyklusdurchlauf aktiviert.

Wenn Kontakt 10001 stromführend wird, beginnt die Funktion den Vergleich bei Matrixposition 1 beim nächsten Pegelwechsel von 10001 und hält wieder an, wenn der Wert in Register 44622 = 17 ist. Wenn Kontakt 10002 nicht stromführend ist, wird die Funktion beim nächsten Pegelwechsel von 10001 den Vergleich bei Matrixposition 18 beginnen und anhalten, wenn der Wert in Register 44622 = 25 ist.

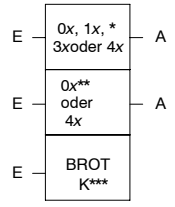
Bit-Abfrage und -Veränderung in einer Datenmatrix

Es stehen drei Befehle für die Abfrage und Veränderung der Bitmuster in einer Datenmatrix zur Verfügung:

- □ Der Bit-Abfrage-Befehl (SENS) prüft und berichtet den Inhalt - 1 oder 0 - spezifischer Bits in der Matrix
- □ Der Bit-Verändern-Befehl (MBIT) verändert den Inhalt eines spezifischen Bits in einer Matrix - d.h. ändert ein 0 Bit auf 1 und setzt für ein 1 Bit 0 ein
- ❒ Der Bit-Rotations-Befehl verschiebt das Bitmuster in einer Matrix nach links oder rechts und erzwingt dabei, daß das Bit, das herausgeschoben wird, entweder aus der Matrix fällt oder am anderen Ende des Registers wieder eingesetzt wird

Mit Hilfe dieser Befehle kann pro Zyklusdurchlauf die Information für ein Bit ermittelt, es verändert oder rotiert werden. Es handelt sich dabei jeweils um Dreierelement-Befehle.

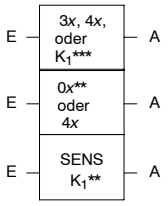
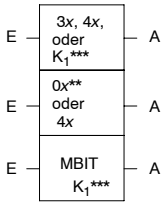
Warnung: MBIT und BROT übergehen alle auf disabled gesetzten Spulen einer Zieltabelle, ohne sie auf enabled zu ändern. Wenn eine Spule aus Reparatur- oder Wartungsgründen auf disabled gesetzt wurde, besteht Verletzungsgefahr, da der Zustand dieser Spule sich durch die Bit-Veränderung verändern kann.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Bit Rotation		Oben: EIN initiiert die Bit-Rotation Mitte: 0 = Beginn links 1 = Beginn rechts Unten: 0 = Bit fällt heraus aus dem Register 1 = Bit bricht um zum Anfang des Registers	Oben: Quellmatrix Mitte: Zielmatrix Unten: Matrixlänge*	Oben: Kopie des oberen Eingangs Mitte: Inhalt des Bits, das aus der Matrix herausrotiert	Rotiert und verschiebt Bitmuster in einer Matrix, pro Zyklus eine Verschiebung

* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß sie ein Mehrfaches von 16 + 1 sein (1, 17, 33, etc.) und impliziert den Gebrauch von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

** Wenn 0*x* Referenzen als Ziel benutzt werden, können diese nicht als Spulen programmiert werden, sondern nur als Kontakte, die auf diese Spulennummern verweisen.

*** K ist eine Integer-Konstante im Bereich 1 ... 100 ;
K_1 ist eine Integer-Konstante im Bereich 1 ... 255

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Bit Abfrage		Oben: ON gibt Inhalt der Matrix- bits wieder Mitte: Inkrementiert den Zeiger nach einer Bit-Inhaltsermittlung Unten: Stellt Zeiger auf 1 zurück	Oben: Zeiger auf die Matrix Mitte: Adresse des ersten Registers in der Matrix Unten: Matrixlänge**	Oben: Kopie des oberen Eingangs Mitte: kopiert das ermittelte Bit Unten: Zeiger > Matrixlänge	Prüft und berichtet den Inhalt eines bestimmten Bits=d.h. 1 oder 0=in einer Matrix; pro Zyklus wird 1 Bit geprüft
Bit Veränderung		Oben: EIN ändert den Inhalt der Bits der Matrix Mitte: 0 = Bit löschen 1 = Bit setzen Unten: Inkrementiert den Zeiger nach einer Bit-Änderung	Oben: Zeiger auf die Matrix Mitte: Adresse des erstenRegisters in der Matrix Unten: Matrixlänge**	Oben: Kopie des oberen Eingangs Mitte: Kopie des mittleren Eingangs Unten: Zeiger > Matrixlänge	Ändert den Wert eines Bits in der Matrix von 0 zu 1 oder von 1 zu 0; pro Zyklus kann 1 Bit geändert werden

* Wenn Sie eine 0*x* oder 1*x* Referenz benutzen, muß sie ein Mehrfaches von 16 + 1 sein (1, 17, 33, etc.) und impliziert den Gebrauch von 16 diskreten Bits (1 ... 16, 17 ... 32, 33 ... 48, etc.).

** Wenn 0*x* Referenzen als Ziel benutzt werden, können diese nicht als Spulen programmiert werden, sondern nur als Kontakte, die auf diese Spulennummern verweisen.

*** K ist eine Integer-Konstante im Bereich 1 ... 100 ;
K_1 ist eine Integer-Konstante im Bereich 1 ... 255

Kapitel 8
Einfache ASCII Kommunikation

- ☐ASCII-Kommunikation mit Hilfe der Kontaktplanlogik
- ☐Der COMM-Befehl
- ☐Datenformate
- ☐ASCII-Zeichensätze
- ☐Anwendungsbeispiel: Benutzung des HHP als ASCII Datenanzeigegerät

ASCII-Kommunikation mit Hilfe der Kontaktplanlogik

Der COMM-Befehl gibt Ihnen die Möglichkeit, Daten von ASCII-Geräten, wie Tastaturen, Displays, Barcode-Lesern, zu lesen oder an diese zu senden, was über einen der an der SPS eingebauten Kommunikations-Ports geschehen kann, oder, wenn die SPS ein Parent ist, über den Comm-Port an einer der Child-SPS innerhalb der Erweiterungskopplung.

Definierte Nachrichtenformate

Die bei den Modicon Micro SPS angebotenen ASCII Kommunikationsmöglichkeiten benutzen einfache, definierte Nachrichtenformate. Auf diese Weise können Sie das kostengünstige Handprogrammiergerät (HHP) 520VPU19200 als ASCII-Gerät einsetzen. Das HHP selbst unterstützt die ASCII Nachrichten-/Format-Bearbeitung nicht in vollem Umfang.

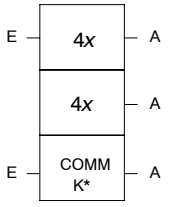
Die untenstehende Tabelle zeigt die Formate, d.h. Operationstypen, die im COMM-Befehl benutzt werden können.

Der Unterschied zwischen den Formaten mit CR/LF und denen ohne CR/LF ist die Art, wie carriage return (Wagenrücklauf) und linefeed (Zeilenvorschub) eingesetzt werden:

- ☐ Bei einem Schreibvorgang mit CR/LF sendet der COMM-Befehl automatisch ein carriage return/linefeed nach der festgelegten Anzahl von Elementen
- ☐ Bei einem Schreibvorgang ohne CR/LF sendet der COMM-Befehl keine carriage return/linefeed Zeichen
- ☐ Bei einem Lesevorgang mit CR/LF wird dem Format Genüge getan, wenn entweder die festgelegte Anzahl der Zeichen eingegeben wurde — d.h. dem Ausgangspuffer entnommen wurde — oder wenn Sie ein carriage return oder linefeed eingeben. Im zweiten Fall wird CR/LF nicht in ein Register gespeichert
- ☐ Bei einem Lesevorgang ohne CR/LF ist die Eingabe der festgelegten Anzahl der Zeichen der einzig mögliche Weg, dem Format zu entsprechen

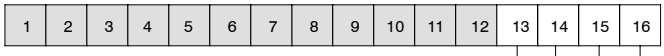
Definierte Nachrichtenformate	
Format	**Dezimalformat-Angabe**
Flush des Eingangs-Puffers	1000
Flush des Eingabebytes, kein CR/LF	1001
Lesen von ASCII-Zeichen, kein CR/LF	1010
Schreiben von ASCII-Zeichen, kein CR/LF	1110
Lesen von ASCII-Zeichen, CR/LF	1020
Schreiben von ASCII-Zeichen, CR/LF	1120
Lesen Integer (1 ... 4), kein CR/LF	1031 ... 1034
Schreiben Integer (1 ... 4), kein CR/LF	1131 ... 1134
Lesen Integer (1 ... 4), CR/LF	1041 ... 1044
Schreiben Integer (1 ... 4), CR/LF	1141 ... 1144
Lesen Hex (1 ... 4), kein CR/LF	1051 ... 1054
Schreiben Hex (1 ... 4), kein CR/LF	1151 ... 1154
Lesen Hex (1 ... 4), CR/LF	1061 ... 1064
Schreiben Hex (1 ... 4), CR/LF	1161 ... 1164

Der COMM-Befehl

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Einfaches ASCII Lesen/ Schreiben		Oben: EIN ruft die comm Funktion auf Unten: Abbruch deraktiven Funktion setzen des Ausgangs in der Mitte	Oben: Anfang des Steuerblocks Mitte: Lesefunktion Quelle oder Schreibfunktion Ziel Unten: Größe der Quell-/ Zieltabelle	Oben: AKTIVER Ausgang Mitte: Für einen Zyklus EIN, wenn ein Fehler festgestellt wird Unten: Für einen Zyklus EIN,wenn Funktion beendet ist	Führt die ASCII Kommunikations funktion aus, die im ersten Register des Steuerblocks definiert ist (Register 4*x* in dem oberen Element)
*K ist eine Integer-Konstante im Bereich 1 ... 255					

COMM Steuerblock (auf den durch das Register im obersten Element des Befehls gezeigt wird)

Registernummer	Registerinhalt
4*x*	Definiertes Nachrichtenformat (einer der dezimalen Formatverweiszahlen der Tabelle auf der vorherigen Seite)
4*x* + 1	**COMM Fehlerstatus**



Kein Fehler	0 0 0 0	00
Unkonfiguriertes Child in Register 4*x* + 5	0 0 0 1	01
COMM-Befehl länger aktiv als in Register 4*x* + 9 festgelegte Zeit	0 0 1 0	02
Ungültige Operationsart (Format) in Register 4*x ausgewählt*	0 0 1 1	03
Anzahl der in Register 4*x* + 2 spezifizierten Datenfelder größer als die Konstante im untersten Element des COMM-Befehls	0 1 0 0	04
Fehler im Empfangspuffer	0 1 0 1	05
Ungültiger Integerwert in ein- oder ausgehenden Daten festgestellt	0 1 1 0	06
Ungültiger Hexwert in ein- oder ausgehenden Daten festgestellt	0 1 1 1	07
Anzahl der zu sendenden Bytes überschreitet Größe des Sendepuffers — 256 Bytes für den lokalen ASCII-Port, 64 Bytes für jedes Child	1 0 0 0	08
Kein lokaler ASCII-Port konfiguriert	1 0 0 1	09
Port bereits von Parent/Child benutzt	1 0 1 0	10
Child nicht OK	1 0 1 1	11
DSR-Verbindung aktiv	1 1 0 0	12

Hinweis Die Tabelle auf der nächsten Seite enthält weitere Details und Handlungsanweisungen für aufgetretene Fehler

4*x* + 2	Anzahl vorhandener/erwarteter Datenfelder
4*x* + 3	Anzahl verarbeiteter Datenfelder (Dieses Register wird von dem Befehl verwaltet)
4*x* + 4	reserviert für Modicon
4*x* + 5	Port-Nummer (1 für einen Port an der lokalen SPS, 2 ... 5 wenn die lokale SPS ein Parent ist, die den Port eines Child nutzt)
4*x* + 6	reserviert für Modicon
4*x* + 7	reserviert für Modicon
4*x* + 8	reserviert für Modicon
4*x* + 9	Aktiver Statustimer

COMM Fehlermeldungen (enthalten im zweiten Wort des Steuerblocks)		
Code	**Fehler**	**Erläuterungen**
01	Unkonfiguriertes Child in Register 4*x* + 5 gewählt	Der Wert in Register 4*x* + 5 spezifiziert, mit welcher SPS der COMM-Befehl kommunizieren soll. Der Wert 1 wählt die lokale SPS aus, der Wert 2 das Child #1, der Wert 3 das Child #2 usw. Hinweis Der physikalische Port, der an der gewählten SPS für ASCII Kommunikation genutzt werden soll — comm1, comm2, oder exp link — wird bei der Konfiguriation festgelegt. Er kann nicht dynamisch mit dem COMM-Befehl festgelegt werden.
02	COMM-Befehl länger aktiv als in Register 4*x* + 9 festgelegte Zeit	
03	Ungültige Operationsart (Format) in Register 4*x ausgewählt*	
04	Anzahl der in Register 4*x* + 2 spezifizierten Datenfelder größer als die Konstante im untersten Element des COMM-Befehls	Bei ASCII-Formaten kann jedes Register zwei Felder (ASCII Zeichen) enthalten. Bei der Größe 1 muß daher die Anzahl der Felder$\leq$ 2; bei der Größe 2 muß die Anzahl der Felder $\leq$ 4 sein; etc. Bei Integer-Formaten (1 ... 4) kann jedes Register ein Feld (mit einer Integer-Zahl) mit einer Breite von 1 ... 4 Ziffern enthalten. Bei Hexadezimal-Formaten (1 ... 4) kann jedes Register ein Feld (mit einer Hexzahl) mit einer Breite von 1 ... 4 4-Bit-Einheiten enthalten. Bei allen Formaten, die ein return oder line feed hinzufügen, belegt das return/line feed keinen zusätzlichen Speicherplatz.
05	Fehler im Empfangspuffer	Es kann sich um einen Paritäts-, Überlauf-, oder Blockfehler handeln. Zum Löschen des Fehlers muß der Puffer gelöscht werden.
06	Ungültiger Integerwert in ein- oder ausgehenden Daten festgestellt	Gültige Werte (in Dezimalformat): Für I1 0 ... 9 Für I2 0 ... 99 Für I3 0 ... 999 Für I4 0 ... 9999
07	Ungültiger Hexwert in ein- oder ausgehenden Daten festgestellt	Gültige Hexwerte: Für H1 0 ... F Für H2 0 ... FF Für H3 0 ... FFF Für H4 0 ... FFFF

COMM Fehlermeldungen (enthalten im zweiten Wort des Steuerblocks)		
Code	**Fehler**	**Erläuterungen**
08	Anzahl der zu sendenden Bytes überschreitet Größe des Sendepuffers=256 Bytes für den lokalen ASCII-Port, 64 Bytes für jedes Child	Die Anzahl der zu versendenden Bytes ist abhängig vom gewählten Format und der Anzahl der Felder, die verarbeitet werden sollen. Beim ASCII-Format gilt: Anzahl der Bytes = Anzahl der zu verarbeiteten Formate. Bei Integer- und Hexformaten: Anzahl der Bytes = Anzahl der zu verarbeiteten Formate x Formatbezeichner (1 ... 4). Wenn z.B. die Anzahl der zu verarbeitenden Felder 2 und der Formatbezeichner 3 ist , beträgt die Anzahl der zu sendenden Bytes 6 (2x3). Zu allen Formaten muß die Zahl 2 addiert werden, wenn ein return/line feed gewählt wird=returns und line feed werden im TX-Puffer gespeichert.
09	Kein lokaler ASCII-Port konfiguriert	Konfigurieren Sie die SPS neu und weisen Sie den gewünschten ASCII-Port zu
10	Port bereits von Parent/Child benutzt	Bei einem Parent zeigt dieser Fehler an, daß die Einheit versucht, auf den ASCII-Port eines Child zuzugreifen, wobei dieser Port für die Benutzung durch das Child selbst konfiguriert wurde. Konfigurieren Sie das Child neu und weisen Sie den gewünschten Port dem Parent zu. Bei einem Child zeigt dieser Fehler an, daß die Einheit versucht, auf den lokalen ASCII-Port zuzugreifen, wobei dieser Port für die Benutzung durch den Parent konfiguriert wurde. Konfigurieren Sie das Child neu und weisen Sie den gewünschten Port dem Child zu.
11	Child nicht OK	Der Parent ist nicht in der Lage, mit dem Child über die Erweiterungskopplung zu kommunizieren.
12	DSR-Verbindung aktiv	Obwohl ein Comm-Port für ASCII konfiguriert wurde, kann er in Wirklichkeit im Modbus/ASCII-Wechselmodus sein, wobei die DSR-Verbindung für die Umschaltung zwischen den beiden Kommunikationsprotokollen benutzt wird. Wenn die SPS gestoppt wird oder die DSR-Verbindung während des Laufs der SPS erneut definiert wird, wird der Port mit den in der Konfiguration festgelegten Modbus-Parametern neu konfiguriert. Wenn die SPS läuft und die DSR-Verbindung während des Laufs der SPS nicht erneut definiert wird, wird der Port mit den in der Konfiguration festgelegten ASCII-Parametern neu konfiguriert.

Datenformate

ASCII Zeichenformat

Formatzahlen	1010, 1110, 1020, 1120
Allgemeiner Gebrauch	Senden/Empfangen von ASCII-Zeichen oder 8-Bit-Daten. Die Daten sind in zwei Zeichen pro 4*x* Register gepackt, das erste Zeichen in den signifikantesten acht Bits des Registers und das zweite Zeichen in den am wenigsten signifikanten acht Bits
Benutzung bei einem Schreibvorgang	
Kein Auto CR/LF	Format erfüllt nach Ausgabe von *n* Zeichen aus Register
Auto CR/LF	Format erfüllt nach Ausgabe von *n* Zeichen aus Register und Ausgabe von CR/LF
Benutzung bei einem Lesevorgang	
Kein Auto CR/LF	Format erfüllt nach Eingabe von *n* Zeichen in die Register
Auto CR/LF	Format erfüllt nach Eingabe von *n* Zeichen in die Register oder Empfang von CR/LF im Puffer

Integerformat (1 ... 4)

Formatbezeichner	1031 ... 1034, 1131 ... 1134, 1041 ... 1044, 1141 ... 1144
Allgemeine Benutzung	Senden/Empfangen von Integer-Datenfeldern. Die Daten sind in Form von 1 ... 4 Ziffern pro 4*x* Register gepackt (abhängig von dem gewählten Formatbezeichner) und rechtsbündig ausgerichtet mit der ersten, ganz links stehenden Ziffer des Datenfelds

Benutzung bei einem Schreibvorgang

Kein Auto CR/LF	Format erfüllt nach Ausgabe von *n* Datenfeldern der Register
Auto CR/LF	Format erfüllt nach Ausgabe von *n* Datenfeldern der Register und Ausgabe von CR/LF

Benutzung bei einem Lesevorgang

Kein Auto CR/LF	Format erfüllt nach Eingabe von *n* Integer-Zahlen in die Register
Auto CR/LF	Format erfüllt nach Eingabe von *n* Integer-Zahlen in die Register oder Empfang von CR/LF im Puffer

Hex-Format (1 ... 4)

Formatbezeichner	1051 ... 1054, 1151 ... 1154, 1061 ... 1064, 1161 ... 1164
Allgemeine Benutzung	Senden/Empfangen von hexadezimal-Datenfeldern. Die Daten sind in Form von 1 ... 4 Ziffern pro 4*x* Register gepackt (abhängig von dem gewählten Formatbezeichner) und rechtsbündig ausgerichtet mit der ersten, ganz links stehenden Ziffer des Datenfelds

Benutzung bei einem Schreibvorgang

Kein Auto CR/LF	Format erfüllt nach Ausgabe von *n* Datenfeldern der Register
Auto CR/LF	Format erfüllt nach Ausgabe von *n* Datenfeldern der Register und Ausgabe von CR/LF

Benutzung bei einem Lesevorgang

Kein Auto CR/LF	Format erfüllt nach Eingabe von *n* Integer-Zahlen in die Register
Auto CR/LF	Format erfüllt nach Eingabe von *n* Integer-Zahlen in die Register oder Empfang von CR/LF im Puffer

Flush (Löschen) des Eingabepuffer-Formats

Formatbezeichner	1000
Allgemeine Benutzung	Löschen des Eingabepuffers. In der lokalen SPS wird der Puffer sofort, d.h. in der Logikbearbeitungszeit, gelöscht. Wenn ein Parent den Comm-Port eines Child für ASCII-Operationen benutzt, wird die Löschung durchgeführt wenn das Child die Aufforderung dazu von dem Parent erhält — der Parent sendet diese Aufforderung am Ende des Zyklusdurchlaufs

Benutzung bei einem Schreibvorgang

Trifft nicht zu

Benutzung bei einem Lesevorgang

Alle Bytes im Eingabepuffer werden gelöscht

Flush (Löschen) des Eingabebyte-Formats

Formatbezeichner	1001
Allgemeine Benutzung	Löschen einer Anzahl von Bytes im Eingabepuffer. In der lokalen SPS werden die Bytes sofort gelöscht. Wenn eine Mutter den Comm-Port eines Child für ASCII-Operationen benutzt, wird die Löschung durchgeführt, wenn das Child die Aufforderung dazu von dem Parent erhält — der Parent sendet diese Aufforderung am Ende des Zyklusdurchlaufs

Benutzung bei einem Schreibvorgang

Trifft nicht zu

Benutzung bei einem Lesevorgang

Die angegebene Anzahl der Bytes im Eingabepuffer werden gelöscht

ASCII Zeichensatz

Dezimal	Oktal	Hex	ASCII-Zeichen	Dezimal	Oktal	Hex	ASCII Symbol
0	000	00	NUL (null)	33	041	21	!
1	001	01	SOH (Kopfanfang)	34	042	22	″
2	002	02	STX (Textanfang)	35	043	23	#
3	003	03	ETX (Textende)	36	044	24	$
4	004	04	EOT (Übertragungsende)	37	045	25	%
5	005	05	ENQ (Stationsaufforderung)	38	046	26	&
6	006	06	ACK (Bestätigung)	39	047	27	′
7	007	07	BEL (Signalton)	40	050	28	(
8	010	08	BS (Rückschritt)	41	051	29	)
9	011	09	HT (Horizontaler Tab)	42	052	2A	*
10	012	0A	LF (Zeilenvorschub)	43	053	2B	+
11	013	0B	VT (Vertikaler Tab)	44	054	2C	‘
12	014	0C	FF (Seitenvorschub)	45	055	2D	–
13	015	0D	CR (Wagenrücklauf)	46	056	2E	.
14	016	0E	SO (Shift aus / Rotes Farbband)	47	057	2F	/
15	017	0F	SI (Shift ein / Schwarzes Farbband)	48	060	30	0
16	020	10	DLE (Datenübertragungsumschaltung)	**49**	061	**31**	1
17	021	11	DC1 (Gerätesteuerung 1 X-EIN)	50	062	32	2
18	022	12	DC2 (Gerätesteuerung 2 Aux-EIN)	51	063	33	3
19	023	13	DC3 (Gerätesteuerung 3 X-AUS)	52	064	34	4
20	024	14	DC4 (Gerätesteuerung 4 Aux-AUS)	53	065	35	5
21	025	15	NAK (Negative Bestätigung)	54	066	36	6
22	026	16	SYN (Synchronisierung)	55	067	37	7
23	027	17	ETB (Ende des Datenübertragungsblocks)	56	070	38	8
24	030	18	CAN (Cancel)	**57**	**071**	**39**	9
25	031	19	EM (Ende der Aufzeichnung)	58	072	3A	:
26	032	1A	SUB (Substitution)	59	073	3B	;
27	033	1B	ESC (Escape)	60	074	3C	
28	034	1C	FS (Hauptgruppentrennzeichen)	61	075	3D	=
29	035	1D	GS (Gruppentrennzeichen)	62	076	3E	>
30	036	1E	RS (Untergruppentrennzeichen)	63	077	3F	?
31	037	1F	US (Teilgruppentrennzeichen)	64	080	40	@
32	040	20	SP (Leertaste)				

Dezimal	Oktal	Hex	ASCII Symbol		Dezimal	Oktal	Hex	ASCII Symbol
65	081	41	A		97	121	61	a
66	082	42	B		98	122	62	b
67	083	43	C		99	123	63	c
68	084	44	D		100	124	64	d
69	085	45	E		101	125	65	e
70	086	46	F		102	126	66	f
71	087	47	G		103	127	67	g
72	090	48	H		104	130	68	h
73	091	49	I		105	131	69	i
74	092	4A	J		106	132	6A	j
75	093	4B	K		107	133	6B	k
76	094	4C	L		108	134	6C	l
77	095	4D	M		109	135	6D	m
78	096	4E	N		110	136	6E	n
79	097	4F	O		111	137	6F	o
80	100	50	P		112	140	70	p
81	101	51	Q		113	141	71	q
82	102	52	R		114	142	72	r
83	103	53	S		115	143	73	s
84	104	54	T		116	144	74	t
85	105	55	U		117	145	75	u
86	106	56	V		118	146	76	v
87	107	57	W		119	147	77	w
88	110	58	X		120	150	78	x
89	111	59	Y		121	151	79	y
90	112	5A	Z		122	152	7A	z
91	113	5B	[		123	153	7B	{
92	114	5C	\		124	154	7C	|
93	115	5D	]		125	155	7D	}
94	116	5E	^		126	156	7E	~
95	117	5F	_		127	157	7F	DEL (Löschen)
96	120	60	‘					

Anwendungsbeispiel: Benutzung des HHP als ASCII Datenanzeigegerät

In diesem Beispiel wird ein 520VPU19200 Handprogrammiergerät als ASCII-Display benutzt, auf dem eine Komponentenzählung ausgeführt wird. Die Anwendung benutzt vier verschiedene COMM-Befehle:

- ☐ Das erste COMM schreibt die ASCII-Nachricht **`PART COUNT =`** . Es wird das Format 1110 benutzt, welches ASCII-Zeichen ohne anschließendem Wagenrücklauf und Zeilenvorschub (CR/LF) schreibt.
- ☐ Das zweite COMM schreibt vier Integerzahlen, welche die Komponentenzählung angeben. Es wird das Format 1144 benutzt, das vier Ziffern mit anschließendem CR/LF schreibt
- ☐ Das dritte COMM schreibt die ASCII-Nachricht **`CYCLE TIME =`** . Es wird das Format 1110 benutzt.
- ☐ Das vierte COMM schreibt vier Ziffern, die die Zykluszeit angeben. Es wird das Format 1144 benutzt

Der erste und der dritte COMM-Befehl benutzen den gleichen Steuerblock. Die ersten 10 Register dieses Steuerblocks, 40400 ... 40409, sehen folgendermaßen aus:

Steuerblock für erstes und drittes COMM		
Register-Nummer	**Register-Wert**	**Bedeutung**
40400	1110	Datenformat: Schreiben von ASCII-Zeichen, CR/LF
40401	*nn*	SPS generiert Fehlermeldung, wobei *nn* im Bereich 00 ... 12 ist (00 steht für: keine Fehler)
40402	14	Ein Maximum von 14 Bytes an Information
40403	*nn*	Anzahl der verarbeiteten Datenfelder (*nn* wird von SPS verwaltet)
40404		Reserviert
40405	1	ASCII Kommunikation wird von lokaler SPS verwaltet
40406		Reserviert
40407		Reserviert
40408		Reserviert
40409	0	Kein Zeitabschaltfehler

Die ASCII-Zeichenketten werden in den Registern 40410 ... 40426 des Steuerblocks gespeichert. Die folgende Tabelle zeigt die zwei ASCII-Zeichen jedes Registers mit ihren hexadezimalen Entsprechungen:

Register-Nummer	**LByte** ASCII	**HByte** ASCII	LByte Hex	HByte Hex
40410	**P**	**A**	50	41
40411	**R**	**T**	52	54
40412	^	**C**	00	43
40413	**O**	**U**	4F	55
40414	**N**	**T**	4E	54
40415	^	**=**	00	3D
40416	^	^	00	00
40420	**C**	**Y**	43	59
40421	**C**	**L**	43	4C
40422	**E**	^	45	00
40423	**T**	**E**	54	49
40424	**M**	**E**	4D	45
40425	^	**=**	00	3D
40426	^	^	00	00
^ steht für ein Leerzeichen				

Der zweite und vierte COMM-Befehl benutzen ebenfalls den gleichen Steuerblock Die ersten 10 Register dieses Steuerblocks, 40430 ... 40439, sehen folgendermaßen aus:

Steuerblock für zweites und viertes COMM		
Register-Nummer	**Register-Wert**	**Bedeutung**
40430	1144	Datenformat: Schreiben von vier Integerzahlen, CR/LF
40431	*nn*	SPS generiert Fehlermeldung, wobei *nn* im Bereich 00 ... 12 ist (00 steht für: keine Fehler)
40432	1	Ein Maximum von 2 Bytes an Information
40433	*nn*	Anzahl der verarbeiteten Datenfelder (*nn* wird von SPSverwaltet)
40434		Reserviert
40435	1	ASCII Kommunikation wird von lokaler SPS verwaltet
40436		Reserviert
40437		Reserviert
40438		Reserviert
40439	0	Kein Zeitabschaltfehler

Register 40440 dieses Steuerblocks enthält den aktuellen Zählerstand. Register 40441 enthält die aktuelle Zykluszeit in Sekunden.

In Netzwerk 1 werden die Steuerblöcke für die vier COMM-Befehle eingerichtet. Dies sind die einzigen Register, die Sie ausfüllen müssen.

Netzwerk 2 vervollständigt den ersten und zweiten COMM-Befehl. Nachdem Spule 00126 stromführend geworden ist, druckt das HHP aus:

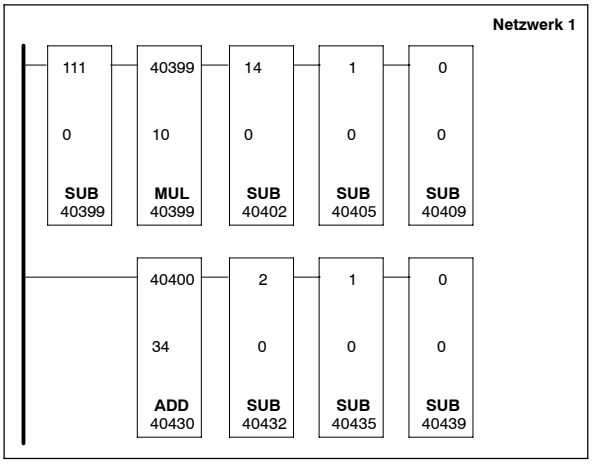
```
PART COUNT = 223
```

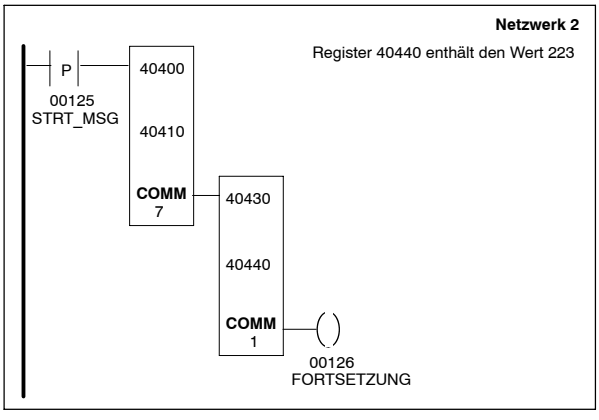
Aufgrund der mit dem Format 1144 verbundenen CR/LF Zeichenfolge, wird der Cursor auf die folgende Zeile bewegt.

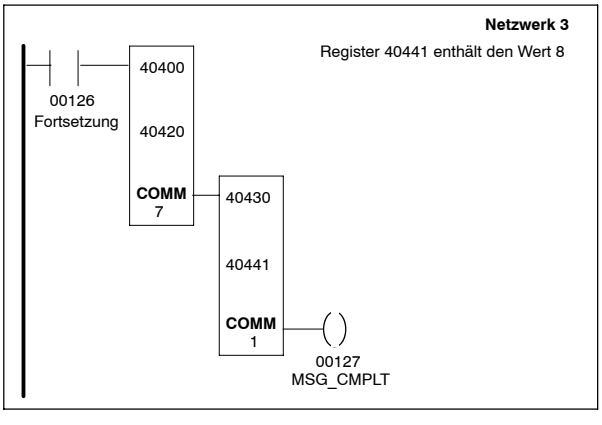
Netzwerk 3 vervollständigt den dritten und vierten COMM-Befehl. Nachdem Spule 00127 stromführend geworden ist, druckt das HHP aus:

```
CYCLE TIME = 8
```

Aufgrund der mit dem Format 1144 verbundenen CR/LF Zeichenfolge, wird der Cursor auf die folgende Zeile bewegt.


Netzwerk 1
111
0
SUB
40399
40399
10
MUL
40399
14
0
SUB
40402
1
0
SUB
40405
0
0
SUB
40409
40400
34
ADD
40430
2
0
SUB
40432
1
0
SUB
40435
0
0
SUB
40439

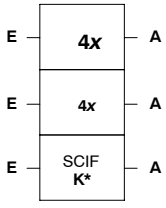

Netzwerk 2
Register 40440 enthält den Wert 223
P
00125
STRT_MSG
40400
40410
COMM
7
40430
40440
COMM
1
00126
FORTSETZUNG


Netzwerk 3
Register 40441 enthält den Wert 8
00126
Fortsetzung
40400
40420
COMM
7
40430
40441
COMM
1
00127
MSG_CMPLT

Kapitel 9
Die Schnittstelle zur Reihenfolgensteuerung

- SCIF-Befehl
- Anwendungsbeispiel: Zeit-Schrittgeber mit SCIF-Blöcken

SCIF-Befehl

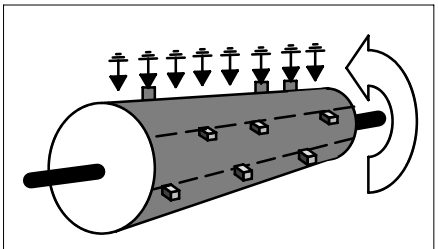
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Reihenfolgen-Steuerung		Oben: EIN führt die Trommel- (Drum) oder ICMP-Operation aus Mitte: EIN im Trommelmodus inkrementiert den Schrittzeiger auf den nächsten Schritt; EIN im ICMP-Modus gibt den Vergleichstatus an den mittleren Ausgang weiter Unten: EIN im Trommelmodus setzt den Schrittzeiger auf 0 zurück; dieser Eingang wird im ICMP-Modus nicht benutzt	Oben: Schrittzeiger Mitte: das erste Register in Schrittdatentabelle. Die ersten sechs Register in der Tabelle sind reserviert, wie unten gezeigt Unten: Anzahl der anwendungsspezifischen Schrittdaten-Register in der Daten-Tabelle. Die Gesamtanzahl der Register der Tabelle beträgt K + 6	Oben: Kopie des oberen Eingangs Mitte: Im Trommelmodus, schaltet EIN beim letzten Schritt – d.h., wenn der Schrittzeiger – maximale Anzahl der-Schritte; EIN im ICMP-modus gibt einen gültigen (1) oder ungültigen (0) Vergleich der Eingänge Unten: EIN , wenn ein Fehler festgestellt wurde	Führt eine von zwei Funktionen aus, wie durch den Wert des ersten Registers in der Schrittdaten- tabelle definiert: 0 = Trommelmodus, bei dem der Block di e Ausgänge in de rTrommelsteueranwendung steuert 1 = Eingangsvergleichs-Modus (ICMP), bei dem der Block die Eingänge liest, um sicherzustellen, das Begrenzungsschalter ,Näherungsschalter ,Drucktasten, etc. richtig positioniert sind, so das die Trommelausgäng e ausgelöst werden können

*K ist eine Integerzahl im Bereich von 1 ... 255.

Register in der Schrittdatentabelle (auf die vom Zeiger im Register des mittleren Elementes		
Referenz	**Registername**	**Beschreibung**
4*x*	Unterfunktion	0 = Trommelmodus Funktionen 1 = Eingangsvergleich-Modus (ICMP) Funktionen (Bei Eingabe anderer Werte in dieses Register werden alle Ausgänge auf AUS gesetzt.)
4*x* + 1	maskierte Ausgangsdaten (im Trommelmodus) Eingangs-Rohdaten (im ICMP-Modus)	Bei jeder Abarbeitung des Blocks durch SCIF geladen; Das Register enthält die Inhalte des *aktuellen Schrittdaten*-Registers maskiert mit dem *Ausgangsmasken*-Register. Vom Anwender aus einer Gruppe aufeinander folgender Eingänge für den Gebrauch im aktuellen Schritt geladen
4*x* + 2	aktuelle Schrittdaten	Bei jeder Abarbeitung des Blocks durch SCIF geladen. Das Register enthält Daten des aktuellen Schritts (auf die der Schrittzeiger zeigt)
4*x* + 3	Ausgangsmaske (im Trommelmodus) Eingangsmaske (im ICMP-Modus)	Vom Anwender vor Gebrauch des Blocks geladen. Der Inhalt wird während der Logikbearbeitung nicht verändert. Enthält eine Maske, die auf die Daten bei jedem einzelnen Schritt angewendet wird Vom Anwender vor Gebrauch des Blocks geladen. Enthält eine Maske, die zu den *Eingangs-Rohdaten* bei jedem Schritt UNDiert wird=maskierte Bits werden nicht verglichen. Die maskierten Daten werden in dem *maskierten Eingangsdaten*-Register abgelegt.
4*x* + 4	im Trommelmodus nicht benutzt maskierte Eingangsdaten (im ICMP-Modus)	 Vom Anwender vor Gebrauch des Blocks geladen. Enthalten das Ergebnis der UNDierten *Eingangsmaske* und der *Eingangs-Rohdaten*
4*x* + 5	Im Trommelmodus nicht benutzt Vergleichsstatus (im ICMP-Modus)	 Vom Anwender vor Gebrauch des Blocks geladen. Enthält das Ergebnis eines XODER der *maskierten Eingangsdaten* und der *aktuellen Schrittdaten*. Unmaskierte Eingänge, die nicht im richtigen logischen Status sind, veranlassen das assoziierte Registerbit, auf 1 zu gehen — Bits, die nicht Null sind, verursachen einen Vergleichsfehler und schalten den mittleren Ausgangs des SCIF-Blocks EIN
4*x* + 6	Beginn der Datentabelle*	Erstes von K Registern der Tabelle mit anwenderspezifischen Steuerdaten
*Dieses und der Rest der Register repräsentieren anwenderspezifische Steuerdaten des zu steuernden Prozesses		

Die Trommelsteuerungs- und die ICMP-Unterfunktionen arbeiten zusammen, um Eingaben zu lesen, Ausgaben zu triggern und Reihenfolgen im Trommelsteuerungsprozess festzulegen. Der SCIF-Befehl emuliert elektronisch eine mechanische Trommelsteuerwalze, wie sie zu Anfang dieses Jahrhunderts aufkamen und auch heute noch bei Anwendungen benutzt werden, bei denen eine gleichzeitige Steuerung mehrerer Motoren, Ventile, Magnetspulen etc. in verschiedenen Schritten des Prozesses erforderlich ist.

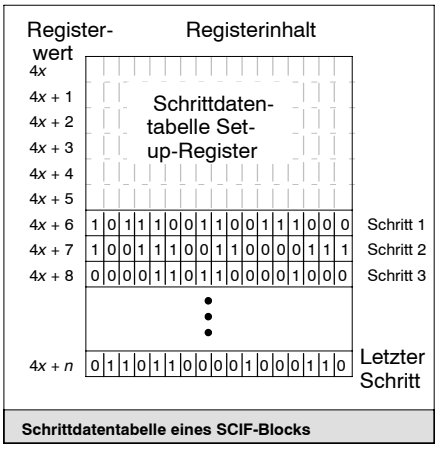
Eine mechanische Trommelsteuerwalze arbeitet ganz ähnlich wie eine Klavierwalze. Ein Zylinder besteht aus mehreren Reihen von Nocken und flachen Oberflächen. Jede Reihe stellt einen Schritt in einem Prozess dar, und jede Nocke bedeutet eine Änderung des Zustands eines mechanischen Geräts innerhalb des Prozesses. Der Zylinder dreht sich nur in eine Richtung, so daß jede Reihe eine fest angebrachte Kontaktleiste passiert, jeweils eine Reihe zur Zeit. Wenn die Nocken in einer bestimmten Reihe auf die Kontakte treffen, werden für diesen Schritt des Prozesses mechanische Zustandsänderungen ausgelöst.



Eine mechanische Trommelsteuerwalze

Bei einem SCIF-Block wird eine Schrittdaten-Tabelle mit einem 16-Bit Register eingerichtet, um so jeden Schritt des zu steuernden Prozessablaufs darzustellen. Die Logik bearbeitet die Tabelle von oben nach unten, wobei der Wert 1 in einem Register wie ein Steuernocken und 0 wie die flache Oberfläche einer mechanischen Trommelsteuerwalze behandelt werden.

Der SCIF-Befehl kombiniert das Konzept der mechanischen Trommelsteuerwalze mit der Mächtigkeit und Flexibilität der Modicon SPS. Dadurch ergeben sich die folgenden Vorteile:


Register-wert
Registerinhalt
4x
4x + 1
4x + 2
4x + 3
4x + 4
4x + 5
Schrittdaten-tabelle Set-up-Register
4x + 6 1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 0 0 Schritt 1
4x + 7 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1 1 Schritt 2
4x + 8 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 0 Schritt 3
4x + n 0 1 1 0 1 1 0 0 0 0 1 0 0 0 1 1 0 Letzter Schritt

Schrittdatentabelle eines SCIF-Blocks

- ☐ Die Ausfallzeiten werden reduziert, da weniger bewegliche Teile erforderlich sind
- ☐ Die Reihenfolgensteuerung kann durch einfache Programmierung und Wartung erfolgen
- ☐ Eine höhere zeitliche Genauigkeit zwischen den Prozess-Schritten
- ☐ Mehr Flexibilität bei der Definierung von Verweil-, Clamping- und Haltezeiten

Moderne Anwendungen für Trommelsteuerungen umfassen Reifen- und Gummiformung, Spritzgußverfahren, Schalenguß, Plattierung, Abziehverfahren und andere, sich aus aufeinanderfolgenden Einzelschritten zusammensetzende Verfahren.

SCIF verbindet zwei Unterfunktionen — Trommelsteuerung und ICMP. Der Trommelsteuerungs-Modus wird benutzt, um ein vorher definiertes Bitmuster den Ausgängen einer Modicon SPS in einer Schritt-für-Schritt aufeinanderfolgenden Reihefolge zuzuordnen. Der ICMP-Modus (Eingabevergleich) wird dazu benutzt, die Eingänge, die von den Feldgeräten kommen, einer vorher definierten Bitmuster-Tabelle für jeden Schritt der Trommelsteuerung zuzuordnen.

Der gemeinsame Gebrauch von Trommelsteuerung und ICMP ermöglicht dem Programmierer, Ausgänge zu bestimmten Zeiten auszulösen und den Status der Eingänge mit einem vorgegebenen Status zu vergleichen. Der Vorgang wird angehalten, wenn eine falsche Zuordnung auftritt.

Anwendungsbeispiel: Zeit-Schrittgeber mit SCIF-Blöcken

Dieses Beispiel einer aus drei Netzwerken bestehenden Kontaktplanlogik-Anwendung, zeigt, wie SCIF Blöcke sowohl im Trommelsteuerungs- als auch ICMP-Modus benutzt werden können. Die Logik in Netzwerk 1 startet und stoppt den Ablaufschrittzyklus. Wenn die Drucktaste *StartZykl* gedrückt wird, arbeitet die Logik die Trommelsteuerung zyklisch ab, bis entweder die Taste *StopZyklus* oder *Not-Stop* betätigt wird.

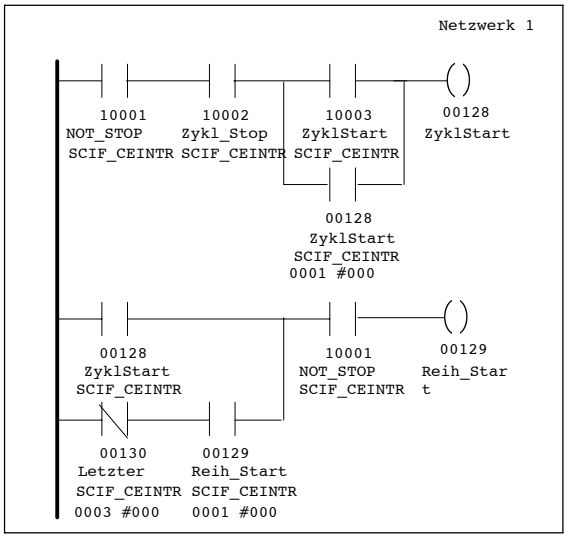
Wenn *StopZyklus* aufgerufen wird, wird die Trommelsteuerung so lange weiter abgearbeitet, bis der letzte Schritt der Schritt-Tabelle vollendet ist. Wird *Not-Stop* betätigt, stoppt die Trommelsteuerung sofort auf dem aktuellen Schritt.

☞ **Hinweis:** Bei einigen Anwendungen kann die Implementierung eines*Not-Stop* unerwünscht sein. Wenn in Ihrer Anwendung ein sofortiger Stop auf dem aktuellen Schritt nicht erwünscht ist, sollten Sie die Logik gemäß Ihren speziellen Anforderungen ändern.

Netzwerk 1 steuert den Start und Stop des Trommelsteuerungs-Beispiels.

Spule 00128=Zyklusstart SCIF_CEINTR= zeigt an, daß der SCIF-Zyklus begonnen hat.

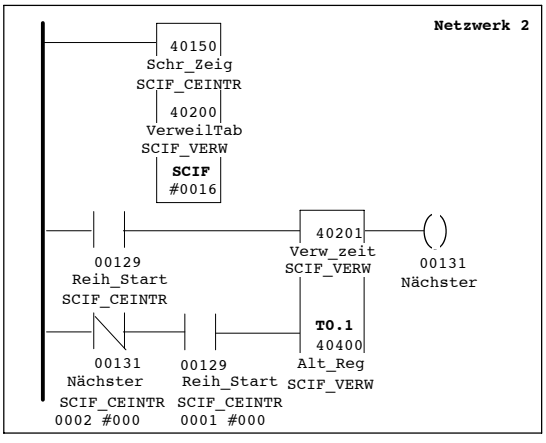
Spule 00129=Reih_Start SCIF_CEINTR=zeigt an, daß die SCIF-Reihenfolge begonnen hat oder neu gestartet wurde.



Achtung: Benutzen Sie dieses Beispiel nur als Simulation, nicht an Maschinen. Die Ausführung dieses Beispiels erzeugt stromführende Ausgänge.

Netzwerk 2 steuert die Verweilzeit bei jedem Schritt der Trommelsteuerung.

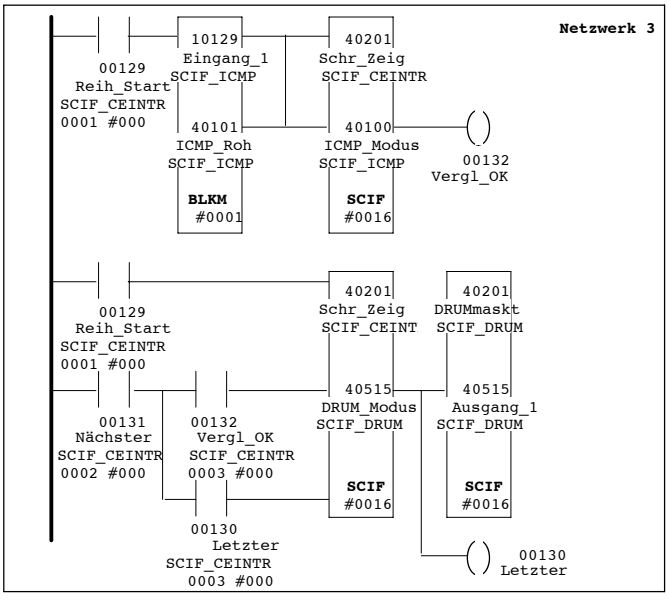
Spule 00131=Nächster SCIF_CEINTR= inkrementiert den SCIF-Zeiger zum nächsten Schritt.



Netzwerk 3 enthält die ICMP- und Trommelsteuerungs-Funktionen, die für den Vergleich der Systemeingänge mit den vorgegebenen Werten und für das Auslösen der Ausgänge der Trommelsteuerung benötigt werden.

Der BLKM-Block in Netzwerk 3 verschiebt die Eingangs-Rückmeldungen, die der benachbarte SCIF-Block im ICMP-Modus gibt, in sein mittleres Registerelement. Dieser SCIF-Block vergleicht anschließend den Status der Rückmeldungsergebnisse mit dem erwarteten Resultat.

Spule 00132=Vergl_OK SCIF_CEINTR=zeigt an, daß die SCIF ICMP Eingänge der gewünschen Vorgabe entsprechen.



Netzwerk 3 führt die eigentiche Reihenfolgesteuerung durch. Bei der Ausführung jedes Schrittes wird Register 40301 durch den Trommelsteuerungs-Modus SCIF-Block geändert und spiegelt das Bitmuster des aktuellen Schrittes wieder.

Der BLKM-Block nimmt die in Register 40301 maskierten Daten und verschiebt sie in die Spulen 00001 ... 00017. Diese Spulen können ggf. direkt realen Ausgängen in einem E/A-Verzeichnis zugeordnet werden, es ist aber ebensogut möglich, daß Kontakte dieser Spulen benutzt werden, um die für die Einschaltung realer Eingänge verantwortliche Logik zu beeinflussen.

Referenzmarkierungen für das Anwendungsbeispiel

Die Referenzen in der untenstehenden Tabelle werden für das Starten, Stoppen und Ineinandergreifen der SCIF-Funktion benutzt:

Steuerreferenzen

Ref #	Symbol	Funktion	Beschreibung
00128	ZyklStart	SCIF_CEINTR	Gibt an, daß der SCIF-Zyklus begonnen hat
00129	Reih_Start	SCIF_CEINTR	gibt an, daß die SCIF-Reihenfolge gestartet/neu gestartet wurde
00130	Letzter	SCIF_CEINTR	gibt SCIF als letzten Schritt an
00131	Nächster	SCIF_CEINTR	inkrementiert den SCIF-Zeiger auf den nächsten Schritt
00132	Vergl_OK	SCIF_CEINTR	gibt an, daß SCIF ICMP Eingänge = gewünschte Vorgaben
10001	NOT_STOP	SCIF_CEINTR	NotStop hält SCIF auf dem aktuellen Schritt an
10002	Stop_Zykl	SCIF_CEINTR	Stop SCIF-Zyklus hält SCIF am Ende des Zyklus an
10003	StartZykl	SCIF_CEINTR	Start startet den SCIF-Zyklus
40150	Schr_Zeig	SCIF_CEINTR	das Schrittzeiger-Register enthält den aktuellen SCIF-Schritt #

Die Referenzen in der untenstehenden Tabelle werden in der SCIF-Verweilfunktion benutzt. Wenn die SCIF-Funktion benutzt wird, um Schritt-Verweilzeiten aufzunehmen, sollte dies im Trommelsteuerungs-Modus = 0 geschehen.

Verweil-Referenzen			
Ref #	**Symbol**	**Funktion**	**Beschreibung**
40400	Alt_Reg	SCIF_VERW	Alt-Register für den Verweil-Timer
40200	Verweiltab	SCIF_VERW	SCIF wird benutzt, um Verweilzeiten für jedem Schritt zu speichern
40201	Ver_zeit	SCIF_VERW	Aktuelle Verweilzeit für aktuellen Schritt
40206	Verweil1	SCIF_VERW	Verweilzeit Schritt 1
40207	Verweil2	SCIF_VERW	Verweilzeit Schritt 2
40208	Verweil3	SCIF_VERW	Verweilzeit Schritt 3
40209	Verweil4	SCIF_VERW	Verweilzeit Schritt 4
40210	Verweil5	SCIF_VERW	Verweilzeit Schritt 5
40211	Verweil6	SCIF_VERW	Verweilzeit Schritt 6
40212	Verweil7	SCIF_VERW	Verweilzeit Schritt 7
40213	Verweil8	SCIF_VERW	Verweilzeit Schritt 8
40214	Verweil9	SCIF_VERW	Verweilzeit Schritt 9
40215	Verweil10	SCIF_VERW	Verweilzeit Schritt 10
40216	Verweil11	SCIF_VERW	Verweilzeit Schritt 11
40217	Verweil12	SCIF_VERW	Verweilzeit Schritt 12
40218	Verweil13	SCIF_VERW	Verweilzeit Schritt 13
40219	Verweil14	SCIF_VERW	Verweilzeit Schritt 14
40220	Verweil15	SCIF_VERW	Verweilzeit Schritt 15
40221	Verweil16	SCIF_VERW	Verweilzeit Schritt 16

Die Referenzen in der untenstehenden Tabelle werden in der SCIF-Eingangsvergleich-Funktion (ICMP) benutzt und geben die Systemeingänge wieder.

ICMP Funktions-Referenzen			
Ref #	**Symbol**	**Funktion**	**Beschreibung**
10017	Eingang_1	SCIF_ICMP	1. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10018	Eingang_2	SCIF_ICMP	2.physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10019	Eingang_3	SCIF_ICMP	3. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10020	Eingang_4	SCIF_ICMP	4.physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10021	Eingang_5	SCIF_ICMP	5. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10022	Eingang_6	SCIF_ICMP	6. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10023	Eingang_7	SCIF_ICMP	7. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10024	Eingang_8	SCIF_ICMP	8. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10025	Eingang_9	SCIF_ICMP	9. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10026	Eingang_10	SCIF_ICMP	10. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10027	Eingang_11	SCIF_ICMP	11. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10028	Eingang_12	SCIF_ICMP	12. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10029	Eingang_13	SCIF_ICMP	13. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10030	Eingang_14	SCIF_ICMP	14. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10031	Eingang_15	SCIF_ICMP	15. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
10032	Eingang_16	SCIF_ICMP	16. physikalischer Eingangsblock-verschoben nach SCIF_ICMP
40100	ICMP_Modus	SCIF_ICMP	Wählt SCIF-Modus Einstellung 1 für ICMP
40101	ICMP_Roh	SCIF_ICMP	Rohdaten-Eingangsregister für SCIF ICMP
40102	ICMP_CSD	SCIF_ICMP	Enthält aktuelle Schrittdaten für die ICMP-Funktion
40103	ICMP_EMask	SCIF_ICMP	Enthält die ICMP Eingangsmaske
40104	ICMPmaskt	SCIF_ICMP	UNDiertes Ergebnis der Rohdaten und der maskierten ICMP-Daten
40105	ICMPstatus	SCIF_ICMP	Enthält XODER der maskierten Daten und ICMP-Schrittdaten
40106	ICMPSchr1	SCIF_ICMP	1. Eintrag in ICMP Datentabelle
40107	ICMPSchr2	SCIF_ICMP	2. Eintrag in ICMP Datentabelle
40108	ICMPSchr3	SCIF_ICMP	3. Eintrag in ICMP Datentabelle
40109	ICMPSchr4	SCIF_ICMP	4. Eintrag in ICMP Datentabelle
40110	ICMPSchr5	SCIF_ICMP	5. Eintrag in ICMP Datentabelle
40111	ICMPSchr6	SCIF_ICMP	6. Eintrag in ICMP Datentabelle
40112	ICMPSchr7	SCIF_ICMP	7. Eintrag in ICMP Datentabelle
40113	ICMPSchr8	SCIF_ICMP	8. Eintrag in ICMP Datentabelle
40114	ICMPSchr9	SCIF_ICMP	9. Eintrag in ICMP Datentabelle
40115	ICMPSchr10	SCIF_ICMP	10. Eintrag in ICMP Datentabelle
40116	ICMPSchr11	SCIF_ICMP	11. Eintrag in ICMP Datentabelle
40117	ICMPSchr12	SCIF_ICMP	12. Eintrag in ICMP Datentabelle
40118	ICMPSchr13	SCIF_ICMP	13. Eintrag in ICMP Datentabelle
40119	ICMPSchr14	SCIF_ICMP	14. Eintrag in ICMP Datentabelle
40120	ICMPSchr15	SCIF_ICMP	15. Eintrag in ICMP Datentabelle
40121	ICMPSchr16	SCIF_ICMP	16. Eintrag in ICMP Datentabelle

Die Referenzen in der untenstehenden Tabelle werden in der SCIF-Trommelsteuerungs-Funktion benutzt und geben die Systemausgänge wieder.

Drum-Referenzen			
Ref #	**Symbol**	**Funktion**	**Beschreibung**
40300	DRUM_Modus	SCIF_DRUM	Wählt SCIF-Modus, Einstellung 0 für Trommel (Drum)
40301	DRUMmaskt	SCIF_DRUM	Maskierter Drumausgang = Maskierte UND aktuelle Schrittdaten
40302	DRUM_CSD	SCIF_DRUM	Trommel aktuelle Schrittdaten (CSD)
40303	DRUM_AMask	SCIF_DRUM	Trommel Ausgangsmaske
40304	DRUM_R1	SCIF_DRUM	Reserviertes Trommel-Register 1
40305	DRUM_R2	SCIF_DRUM	Reserviertes Trommel-Register 2
40306	DRUMSchr1	SCIF_DRUM	1. Eintrag in Trommel Datentabelle
40307	DRUMSchr2	SCIF_DRUM	2. Eintrag in Trommel Datentabelle
40308	DRUMSchr3	SCIF_DRUM	3. Eintrag in Trommel Datentabelle
40309	DRUMSchr4	SCIF_DRUM	4. Eintrag in Trommel Datentabelle
40310	DRUMSchr5	SCIF_DRUM	5. Eintrag in Trommel Datentabelle
40311	DRUMSchr6	SCIF_DRUM	6. Eintrag in Trommel Datentabelle
40312	DRUMSchr7	SCIF_DRUM	7. Eintrag in Trommel Datentabelle
40313	DRUMSchr8	SCIF_DRUM	8. Eintrag in Trommel Datentabelle
40314	DRUMSchr9	SCIF_DRUM	9. Eintrag in Trommel Datentabelle
40315	DRUMSchr10	SCIF_DRUM	10. Eintrag in Trommel Datentabelle
40316	DRUMSchr11	SCIF_DRUM	11. Eintrag in Trommel Datentabelle
40317	DRUMSchr12	SCIF_DRUM	12. Eintrag in Trommel Datentabelle
40318	DRUMSchr13	SCIF_DRUM	13. Eintrag in Trommel Datentabelle
40319	DRUMSchr14	SCIF_DRUM	14. Eintrag in Trommel Datentabelle
40320	DRUMSchr15	SCIF_DRUM	15. Eintrag in Trommel Datentabelle
40321	DRUMSchr16	SCIF_DRUM	16. Eintrag in Trommel Datentabelle
00001	Ausgang_1	SCIF_DRUM	1. physikalischer Ausgangsblock-verschoben von SCIF_DRUM 0000
2	Ausgang_2	SCIF_DRUM	2. physikalischer Ausgangsblock-verschoben vonSCIF_DRUM
00003	Ausgang_3	SCIF_DRUM	3. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00004	Ausgang_4	SCIF_DRUM	4. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00005	Ausgang_5	SCIF_DRUM	5. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00006	Ausgang_6	SCIF_DRUM	6. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00007	Ausgang_7	SCIF_DRUM	7. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00008	Ausgang_8	SCIF_DRUM	8. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00009	Ausgang_9	SCIF_DRUM	9. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00010	Ausgang_10	SCIF_DRUM	10. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00011	Ausgang_11	SCIF_DRUM	11. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00012	Ausgang_12	SCIF_DRUM	12. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00013	Ausgang_13	SCIF_DRUM	13. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00014	Ausgang_14	SCIF_DRUM	14. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00015	Ausgang_15	SCIF_DRUM	15. physikalischer Ausgangsblock-verschoben von SCIF_DRUM
00016	Ausgang_16	SCIF_DRUM	16. physikalischer Ausgangsblock-verschoben von SCIF_DRUM

Kapitel 10
Befehle für Unterprogramme

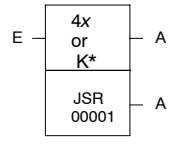
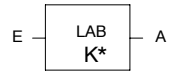
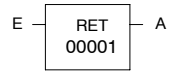
- ☐Befehle für Unterprogramme in der Kontaktplanlogik
- ☐Die Interrupt- und Zähler-/Timer-Eingänge
- ☐Der CTIF-Befehl
- ☐Anwendungsbeispiel eines CTIF-Befehls

Befehle für Unterprogramme in der Kontaktplanlogik

Unterprogramm-Logik kann entweder durch einen Hardware-Interrupt oder durch einen im Programm verankerten Befehl (**JSR**) der Steuerlogik aufgerufen werden. Wenn Sie einen Hardware-Interrupt für die Auslösung des Unterprogramms benutzen, muß der Hochgeschwindigkeits-Eingangsschaltkrei s der SPS für die Verwaltung der Interrupts mit einem Befehl namens **CTIF** konfiguriert werden.

In diesem Kapitel werden die beiden Verfahren besprochen, wie man in ein Unterprogramm hineinkommt, bzw. aus diesem herauskommt.

Die Logik von Unterprogrammen befindet sich immer im letzten Segment des Kontaktplanlogik-Programms. Keine andere Logik, außer dem Unterprogramm, wird dort gespeichert. Beim Aufruf eines Unterprogrammes springt die Logikbearbeitung in einen Befehl namens **LAB** in das letzte Segment. Dieser Befehl bezeichnet den Beginn der Logik dieses Unterprogramms. Wenn die Logikbearbeitung im Unterprogramm einen Befehl mit der Bezeichnung **RET** erreicht, springt sie aus diesem Unterprogramm heraus und kehrt zu der vorherigen Position in der Steuerlogik zurück.

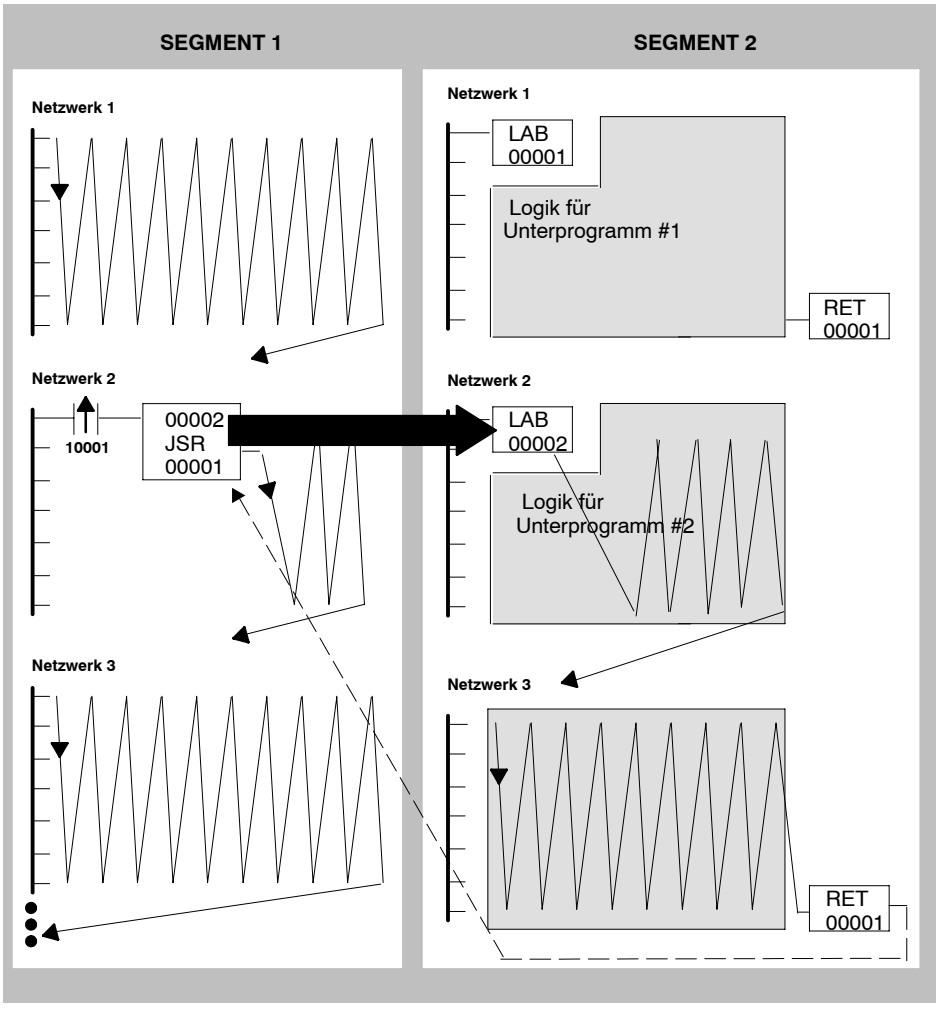
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Springe zu einem Unterprogramm		Oben: EIN setzt das Quell-Unterprogramm auf enabled	Oben: Konstante oder Registerwert, der das gewünschte Unterprogramm angibt Unten: Immer ein konstanter Wert von 1	Oben: Echo des oberen Eingangs Unten: EIN, wenn ein Fehler festgestellt wurde	Veranlaßt Logikzyklus in ein spezifiziertes Unterprogramm im letzten (außerplanmäßigen) Segment der Anwenderlogik zu springen
Markierung des Unterprogramms		Oben: EIN aktiviert das spezifizierte Unterprogramm	Oben: Eindeutiger, konstanter Wert für die Identi-fizierung des gewählten Unterprogramms	Oben: EIN, wenn ein Fehlerfestgestellt wurde	Markiert den Anfangspunkt im Unterprogramm des Anwenderlogik-Segments
Zurück zur Kontaktplanlogik		Oben: EIN inititiiert die Rückkehr aus der Unterfunktion	Oben: Immer ein konstanter Wert von 1	Oben: EIN, wenn ein Fehler festgestellt wurde	Führt Logikzyklus zurück zu dem Elemente, der direkt auf die Stelle folgt, an der in das Unterprogramm gesprungen wurde

*K ist eine Integer-Konstante im Bereich von 1 ... 255

043502910

Nachfolgend ist eine schematische Darstellung abgebildet, die zeigt, wie ein Unterprogramm aus der Kontaktplanlogik aufgerufen wird. Wenn die Logikbearbeitung in Segment 1 auf einen **JSR**-Befehl trifft, der auf enabled gesetzt ist, springt sie zu dem angegebenen Unterprogramm in Segment 2. Nur die zu dem Unterprogramm gehörende Logik wird in Segment 2 ausgeführt — die Logik aller anderen Unterprogramme wird ignoriert.

Wenn die Logikbearbeitung in der Unterprogramm-Logik auf einen **RET**-Befehl trifft, springt sie zurück zu dem Element, das direkt auf den **JSR**-Befehl in Segment 1 folgt.



Die Interrupt- und Zähler-/Timer-Eingänge

Die Modelle 110CPU411, 110CPU512 und 110CPU612 der Micro-SPS sind bereits hardwaremäßig mit einem Satz von Eingabepunkten ausgestattet, die als Hochgeschwindigkeitszähler und/oder Hardwareinterrupts konfiguriert werden können. Diese Eingänge befinden sich auf der linken Seite des Eingangsanschlußblocks an der Oberseite der SPS. (Die einzelnen Anschlußklemmen sind im Hardware-Handbuch Ihrer SPS spezifiziert.)

Diese Eingänge können von der SPS, genau wie Standard-Eingabepunkte, bei jedem Zyklusdurchlauf gelesen werden. Zusätzlich können sie jedoch für die Triggerung von Hochgeschwindigkeitszählern oder hardwaregesteuerten Operationen der Unterprogramme in der Kontaktplanlogik benutzt werden.

Bei der Benutzung als Standard-Eingänge werden diese über die Referenzen 10081 ... 10088 im E/A-Verzeichnis der jeweiligen SPS adressiert. Werden sie zum Triggern von Interrupts oder für Hochgeschwindigkeits-Zähloperationen verwendet, müssen diese Eingänge in der Kontaktplanlogik mit einem Befehl namens CTIF konfiguriert werden. CTIF konfiguriert die interne Hochgeschwindigkeits-Interrupt- und Zähler-Hardware für den Gebrauch dieser Hochgeschwindigkeits-Eingänge. Mit CTIF konfigurierte Eingänge brauchen im E/A-Verzeichnis nicht adressiert zu werden, es sei denn, auf sie bezogene Referenzen werden in dem Kontaktplanlogik-Programm verwendet.

Hardware-Interrupt Operation

Bei der Konfiguration eines Hardware-Interrupts initiiert ein H/L Pegelwechsel im Eingang ein Interrupt-Unterprogramm. Durch Interrupts aufgerufene Unterprogramme sind den durch JSR-Aufruf initiierten Unterprogrammen sehr ähnlich. Sie unterbrechen die normale Logikbearbeitung und leiten sie zu einem LAB-Befehl in Segment 2 weiter, der den Beginn des jeweiligen Unterprogramms kennzeichnet. Das Unterprogramm wird ausgeführt, bis die Logikbearbeitung auf einen RET-Befehl trifft. An diesem Punkt kehrt die Logikbearbeitung zu dem vorherigen Ort in Segment 1 zurück. Der wichtigste Unterschied ist, daß das durch Interrupt ausgelöste Unterprogramm von einem externen Ereignis getriggert wird, gesteuert von einem Gerät, das über Kabel mit dem Eingang verbunden ist, wohingegen das durch JSR initiierte Unterprogramm von internen Bedingungen im Logikprogramm getriggert wird.

Um für den gleichen Eingang mehr als einen Interrupt zu initiieren, muß das Interruptsignal einen niedrigen Wert annehmen und wieder zu einem hohen Wert wechseln. Das Kontaktplanlogik-Betriebssystem läßt keinen neuen Interrupt auf dem gleichen Eingang zu, bevor nicht das vorherige Interrupt-Unterprogramm seit 2 ms beendet ist. Durch diese Verzögerung wird ein Aufhängen der SPS verhindert, das sonst durch einen kontinuierlichen Strom von Hochgeschwindigkeits-Interrupts (> 2 ms) auf dem Eingang verursacht werden könnte.

Der dezidierte Interrupt ist mit der CPU über einen Hardware-Filter verbunden, das zu etwa 50 µs Verzögerung bei dem Interrupt-Unterprogramm führt. Das Betriebssystem läuft bei jedem Zyklusdurchlauf auch für eine gewisse Zeit — etwa 300 µs — mit Interrupts, die auf disabeld gesetzt sind.

Die Initiierung eines Interrupt-Unterprogramms kann daher um etwa 350 µs verzögert werden.

Hochgeschwindigkeits-Eingang	110CPU411				110CPU512				110CPU612	
	00	01	02	03	00	01	02	03	00	03
Dezidierter Interrupt	1	1	1	1	2	1	1	2	2	2
Konfigurierbarer Zähler/ Interrupt	1	1	1	1	1	1	1	1	1	1

Wenn die 110CPU411, 110CPU512, und 110CPU612 SPS wenigstens einen dezidierten Interruptsignal-Eingang haben und einen weiteren Eingang, der mit dem CTIF-Befehl entweder als Hardware-Interrupt oder Hochgeschwindigkeitszähler konfiguriert werden kann:

Der Hochgeschwindigkeitszähler-Eingang

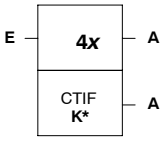
Wenn der konfigurierbare Eingang als Hochgeschwindigkeitszähler eingerichtet wird, muß er mit einem *Zähler-Endwert* konfiguriert und auf enabled gesetzt werden. Dies geschieht mit dem CTIF-Befehl.

Der Zähler zählt die Impulse an seinem Eingang bis der Zähler-Endwert erreicht ist, was den Zählvorgang beendet. Der Eingang kann so konfiguriert werden, daß das Zählereignis einen Interrupt triggert oder Zählerendwert und der aktuelle Zählerstand im E/A-Verzeichnis adressiert werden.

Das Betriebssystem läuft bei jedem Zyklusdurchlauf auch bei auf disabeld gesetzten Interrupts noch für eine gewisse Zeit — etwa 300 µs. Die Initiierung eines Interrupt-Unterprogramms kann daher um etwa 350 µs verzögert werden.

Zur Initiierung eines weiteren Interrupts mit dem gleichen Zähler-Endwert muß der Zähler neu gestartet werden. Das Kontaktplanlogik-Betriebssystem läßt keinen neuen Zähler-Interrupt auf dem gleichen Eingang zu, bevor nicht das vorherige Interrupt-Unterprogramm seit 2 ms beendet ist. Durch diese Verzögerung wird ein Aufhängen der SPS verhindert, das sonst bei einem schnellen Eingangstakt durch einen kleinen Zählerendwert verursacht werden könnte.

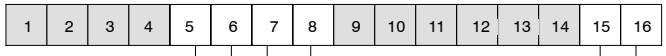
Der CTIF Befehl

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Richtet die Eingänge für Interrupt und Zähler/ Timer Operationen ein		Oben: EIN führt die im oberen Elemente spezi-fizierte Operation aus	Oben: Erstes Wort im CTIF Parameterblock Unten: Nummer der Station, wo die Operation ausgeführt wird	Oben: Kopie des oberen Eingangs Unten: EIN, wenn ein Fehler festgestellt wurde	Konfiguriert die Hardware Interrupts und Zähler/ Timer — endet immer im gleichen Zyklus, in dem er begonnen hat

***K ist eine Integer-Konstante im Bereich 1 ... 5.**

CTIF Parameter-Block

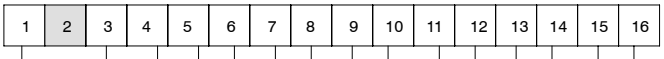
Register 4*x* Fehler/Operations-Typ



0 0 Modus setzen
0 1 Modus feststellen

0 0 0 0	Kein Fehler gefunden
0 0 0 1	Nicht-unterstützten Operatios-Typ angegeben
0 0 1 0	Interrupt 2 wird von diesem Modell nicht unterstützt
0 0 1 1	Interrupt 3 wird nicht unterstützt solange Zähler gewählt
0 1 0 0	Zählerwert von 0 angegeben
0 1 0 1	Zählerwert zu groß (> 16,383)
0 1 1 0	Operatios-Typ wird nur auf lokaler Station unterstützt
0 1 1 1	Spezifizierte Station nicht in E/A-Verzeichnis
1 0 0 0	Kein Unterprogramm für den auf enabled gesetzten Interrupt
1 0 0 1	Remote-Station nicht OK
1 0 1 0	Funktion wird remote nicht unterstützt

Register 4*x* + 1 Steuereinstellung für Operation Modus setzten



Zähler-Endwert laden:
0 Disable
1 Enable

0 1 Zähler-Modus
1 0 Timer-Modus

Disable Interrupt-Service für Int3 0 1
Enable Interrupt-Service für Int3 1 0

0 1 Stop Zähler/Timer Operation
1 0 Start Zähler/Timer Operation

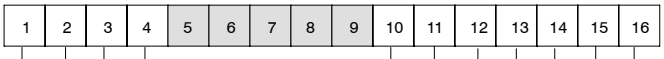
Disable Interrupt-Service für Int2 0 1
Enable Interrupt-Service für Int2 1 0

0 1 Disable Auto-Neustart Operation
1 0 Enable Auto-Neustart Operation

Disable Interrupt-Service für Int1 0 1
Enable Interrupt-Service für Int1 1 0

0 1 Disable Interrupt-Service für Timer/Zähler Eingang
1 0 Enable Interrupt-Service für Timer/Zähler Eingang

Register 4*x* + 2 Status für Operation Modus feststellen



Int3 disabled 0
Int3 enabled 1

0 Zähler-Modus
1 Timer-Modus

Int2 disabled 0
Int2 enabled 1

0 Zähler/Timer Operation gestoppt
1 Zähler/Timer Operation gestartet

Int1 disabled 0
Int1 enabled 1

0 Auto-Neustart Operation disabled
1 Auto-Neustart Operation enabled

Kein Unterprogramm für Timer/Zähler Interrupt

0 Interrupt-Service für Timer/Zähler Eing. disabled
1 Interrupt-Service für Timer/Zähler Eing. enabeled

Kein Unterprogramm für Int1 Interrupt

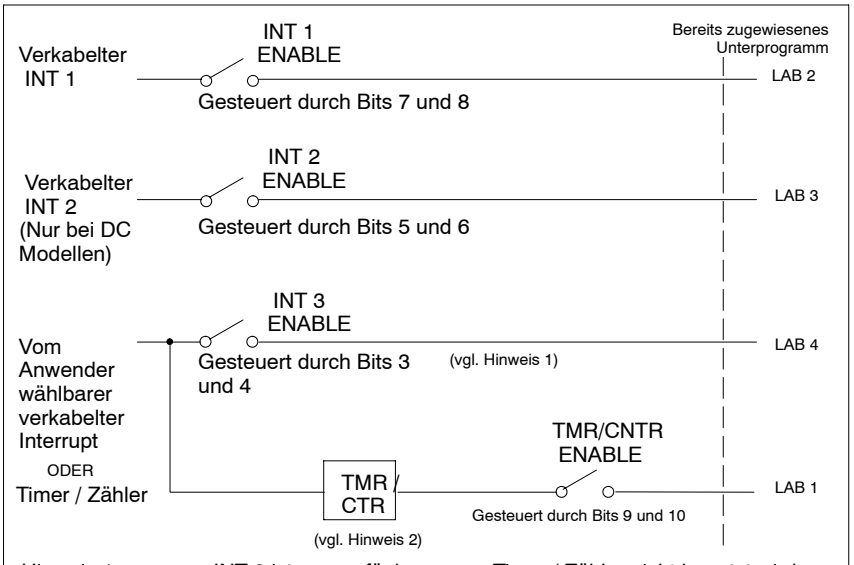
Kein Unterprogramm für Int2 Interrupt

Kein Unterprogramm für Int3 Interrupt

Register 4*x* + 3 Aktueller Zählerwert des Timer/Zähler Eingangs
(durch den Befehlsblock als aktueller Zählerwert im Modus feststellen;
vom Anwender für die Zähler/Timer Vorgabe in Modus setzen festgelegt)

Der CTIF-Befehl ist ein Konfigurations/Operations-Werkzeug für Modicon Micro SPS, die über angeschlossene Interrupts verfügen (alle Modelle außer Modell 110CPU311). Der eigentliche Zähler/Timer und die Interrupts sind Teil der SPS-Hardware, wobei der CTIF-Befehl für die Einstellung dieser Hardware benutzt wird.

Die untenstehende Abbildung zeigt, wie die *Konfigurationsschalter* mit den Interruptfunktionen zusammenhängen.



Hinweis 1. INT 3 ist nur verfügbar, wenn Timer / Zähler nicht benutzt wird.

Hinweis 2. Bits 15 und 16 wählen den Modus (TMR oder CTR). Im CTR-Modus werden die Impulse auf dem Eingang gezählt. Im TMR-Modus arbeitet der Eingang als Timer-Gate und muß für Zeitmessung leitend sein.

Eingangs-Typ	Verfügbar bei 110CPU Modellen	Status-RAM Referenzen für Interruptdaten	Unterprogramm getriggert von diesem Eingang
Vom Anwender wählbarer Timer/ Zähler Interrupt	Alle 411, 512, und 612 Einheiten	10081, einmal pro Zyklus aktualisiert 10084, aktualisiert beim Start jedes Unterprogramms	Unterprogramm #1
Verkabelter Interrupt 1	Alle 411, 512, und 612 Einheiten	10082, einmal pro Zyklus aktualisiert 10085, aktualisiert beim Start jedes Unterprogramms	Unterprogramm #2
Verkabelter Interrupt 2	Nur Einheiten mit DC Stromversorgung	10083, einmal pro Zyklus aktualisiert 10086, aktualisiert beim Start jedes Unterprogramms	Unterprogramm #3
Vom Anwender wählbarer Interrupt 3	Alle 411, 512, und 612 Einheiten	10081, einmal pro Zyklus aktualisiert 10084, aktualisiert beim Start jedes Unterprogramms	Unterprogramm #4

Ein CTIF Anwendungsbeispiel

Das folgende Demonstrationsprogramm bestehend aus sechs Netzwerken, erläutert die Konfiguration der CTIF-Funktion und ihren Gebrauch in den verschiedenen, verfügbaren Modi.

Die ersten beiden Netzwerke, die im Segment 1 des Kontaktplanes stehen, stellen die Steuerlogik dar. Die letzten vier Netzwerke, die in Segment 2 zu finden sind, stellen Logik dar, die aus Unterprogrammen besteht und von angeschlossenen Interrupts aufgerufen wird.

Das Beispiel verdeutlicht:

- den Hochgeschwindigkeits-Zählermodus
- Kombination von Hochgeschwindigkeits-Timer (1 ms) und Interrupt-Modus
- einen diskreten, über Kabel angeschlossenen Interrupt-Modus

In dem Beispiel wird nicht die Fähigkeit gezeigt, zwei verkabelte Interrupts laufen zu lassen, eine Fähigkeit, über die nur die DC SPS verfügen.

Segment 1, Netzwerk 1

Netzwerk 1 des Segments 1, abgebildet auf der folgenden Seite, stellt das erste von zwei Steuernetzwerken dar. Die gesamte, in diesem Netzwerk programmierte, Zähler/Timer-Information wurde in ein E/A-Verzeichnis aufgenommen, damit sie in Register 30001 verfügbar ist. Die Anschluß-Interruptdaten stehen an den Eingängen 10081 ... 10088 zur Verfügung.

Wenn Kontakt 10001 von AUS zu EIN wechselt, wird die Information in den Registern 40501 und 40504 gelöscht. Register 40501, das Sammelregister in Unterprogramm 1 (in Segment 2), wird jedes mal um 1 inkrementiert, wenn es von der Zähler/Timer-Funktion aufgerufen wird. Register 40504, das Sammelregister in Unterprogramm 1 (in Segment 2), wird bei jedem Impuls des hardwaremäßig angeschlossenen Zählers/Timers um 1 inkrementiert.

Die Konfigurationsdaten in den Registern 40100 ... 40103 werden in den CTIF Parameterblock verschoben (in die Register 40300 ... 40303). Diese Information wird sofort an CTIF gesendet und kann ausgeführt werden. Die Information führt zu folgenden Angaben im Parameter-Block:

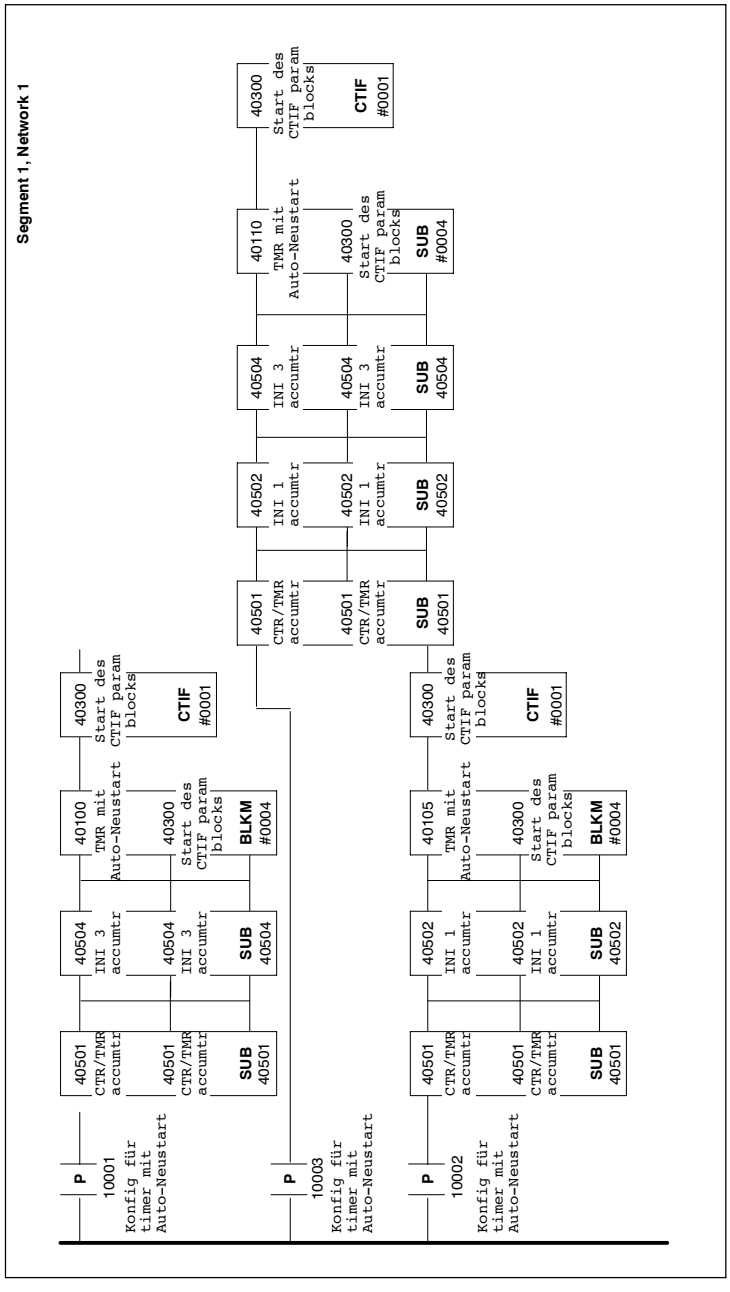
Register	Inhalt
40300	Fehlerkode-Information und Modustyp (immer im Modus Setzen)
40301	Vorliegende Konfigurations-Information wie folgt: Laden des Zählerendwerts enabled Interrupt Service für Int 3 enabled Interrupt Service für Int 2 disabled Interrupt Service für Int 1 disabled Interrupt Service für Int Timer/Zähler Eingang enabled Auto-Neustart Operation enabled Start Timer/Zähler Operation Timer-Modus gewählt Das Bitmuster des Registers ist: 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 (A5AA in Hex)
40302	Staus-Information
40303	Der vorgegebene Wert für den Timer=400

Solange der Hardware-Kontakt die Stellung EIN hat, sammelt der Timer die Eingabedaten. Sobald jedoch der voreingestellte Timerwert erreicht ist, wird Unterprogramm 1 aufgerufen und dessen Funktion ausgeführt, d.h. der Inhalt des Registers 40501 wird um 1 erhöht.

Da die automatische Neustart-Option gewählt wurde, wird der Timer anschließend auf 0 zurückgesetzt und der Timer-Ablauf beginnt erneut, und zwar solange der Hardware-Eingang den Wert EIN hat. Die einzige Bedingung, bei der der Timer sich selbst zurücksetzt, ist das Erreichen des Rücksetzwertes (reset). Interrupt 3 zählt die Anzahl der AUS-EIN-Pegelwechsel des Eingangs.

Bei jedem Pegelwechsel des hardwarmäßig angeschlossenen Timereingangs wird Unterprogramm 4 aufgerufen und seine Funktion ausgeführt, d.h. der Inhalt des Registers 40504 wird um 1 erhöht.

Bei einem Pegelwechsel des Kontakts 10002 von AUS zu EIN wird die Information in den Registern 40501 und 40502 gelöscht. Register 40501, das Sammelregister in Unterprogramm 1 (in Segment 2), wird bei jedem Impuls des hardwaremäßig angeschlossenen Zählers/Timers um 1 inkrementiert.


Segment 1, Network 1
P
10001
Konfig für timer mit Auto-Neustart
40501 CTR/TMR accumtr
40504 INI 3 accumtr
40100 TMR mit Auto-Neustart
40300 Start des CTIF param blocks
CTIF #0001
40501 CTR/TMR accumtr
40504 INI 3 accumtr
40300 Start des CTIF param blocks
BLKM #0004
SUB 40501
SUB 40504
P
10003
Konfig für timer mit Auto-Neustart
40501 CTR/TMR accumtr
40502 INI 1 accumtr
40504 INI 3 accumtr
40110 TMR mit Auto-Neustart
40300 Start des CTIF param blocks
CTIF #0001
40501 CTR/TMR accumtr
40502 INI 1 accumtr
40504 INI 3 accumtr
40300 Start des CTIF param blocks
SUB #0004
SUB 40501
SUB 40502
SUB 40504
P
10002
Konfig für timer mit Auto-Neustart
40501 CTR/TMR accumtr
40502 INI 1 accumtr
40105 TMR mit Auto-Neustart
40300 Start des CTIF param blocks
CTIF #0001
40501 CTR/TMR accumtr
40502 INI 1 accumtr
40300 Start des CTIF param blocks
BLKM #0004
SUB 40501
SUB 40502

Register 40502 ist das Sammelregister in Unterprogramm 2, das bei jedem Impuls des hardwaremäßig angeschlossenen Zählers/Timers um 1 inkrementiert wird.

Die Konfigurationsdaten in den Registern 40105 ... 40108 wird in den CTIF Parameterblock verschoben (Register 40300 ... 40303). Diese Information wird sofort an CTIF gesendet und kann ausgeführt werden. Die Information führt zu folgenden Angaben im Parameter-Block:

Register	Inhalt
40300	Fehlercode-Information und Modus-Typ (immer im Modus Setzen)
40301	Vorhandene Konfigurations-Information wie folgt: Laden des Zählerendstandes enabled Interrupt Service für Int 3 disabled Interrupt Service für Int 2 disabled Interrupt Service für Int 1 enabled Interrupt Service für Int Timer/Zähler Eingang enabled Auto-Neustart Operation enabled Start Timer/Zähler Operation Zählermodus gewählt Das Bitmuster des Registers ist: 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 (96A9 in Hex)
40302	Staus-Information
40303	Der vorgegebene Wert für den Zähler=9999

Der Hardware-Kontakt muß einen Pegelwechsel ausführen, damit der Timer die Zählerwerte weiter sammelt. Sobald der voreingestellte Timerwert erreicht ist, wird Unterprogramm 1 wieder aufgerufen und dessen Funktion ausgeführt, d.h. der Inhalt des Registers 40501 wird bei jedem Aufruf um 1 erhöht.

Da die automatische Neustart-Option gewählt wurde, wird der Timer anschließend auf 0 zurückgesetzt und der Timer-Ablauf beginnt erneut, und zwar solange der Hardware-Eingang den Wert EIN hat. Die einzige Bedingung, bei der der Timer sich selbst zurücksetzt, ist das Erreichen des Rücksetzwertes (reset). Interrupt 3 zählt die Anzahl der AUS-EIN-Pegelwechsel des Eingangs.

Bei jedem Pegelwechsel des angeschlossenen Interrupts von AUS zu EIN wird Unterprogramm 2 aufgerufen und dessen Funktion ausgeführt, d.h. der Inhalt des Registers 40502 wird um 1 erhöht.

Bei einem Pegelwechsel des Kontakts 10003 von AUS zu EIN wird die Information in den Registern 40501 40502 und 40504 gelöscht. Register 40501, das Sammelregister des Unterprogramms, wird bei jedem Aufruf der Zähler-/Timer-Funktion um 1 inkrementiert. Register 40504, das Sammelregister in Unterprogramm 4, wird bei jedem Impuls des hardwaremäßig angeschlossenen Zählers/Timers um 1 inkrementiert.

Die Konfigurationsdaten in den Registern 40110 ... 40113 werden in den CTIF Parameterblock verschoben (in die Register 40300 ... 40303). Diese Information wird sofort an CTIF gesendet und kann ausgeführt werden. Die Information führt zu folgenden Angaben im Parameter-Block:

Register	Inhalt
40300	Fehlercode-Information und Modus-Typ immer im Modus Setzen)
40301	Vorhandene Konfigurations-Information wie folgt: Laden des Zählerendstandes enabled Interrupt Service für Int 3 enabled Interrupt Service für Int 2 disabled Interrupt Service für Int 1 enabled Interrupt Service für Int Timer/Zähler Eingang enabled Auto-Neustart Operation disabled Start Timer/Zähler Operation Zählermodus gewählt Das Bitmuster des Registers ist: 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 (A69A in Hex)
40302	Staus-Information
40303	Der vorgegebene Wert für den Zähler=400

Solange der Hardware-Kontakt die Stellung EIN hat, sammelt der Timer die Eingabedaten. Sobald jedoch der voreingestellte Timerwert erreicht ist, wird

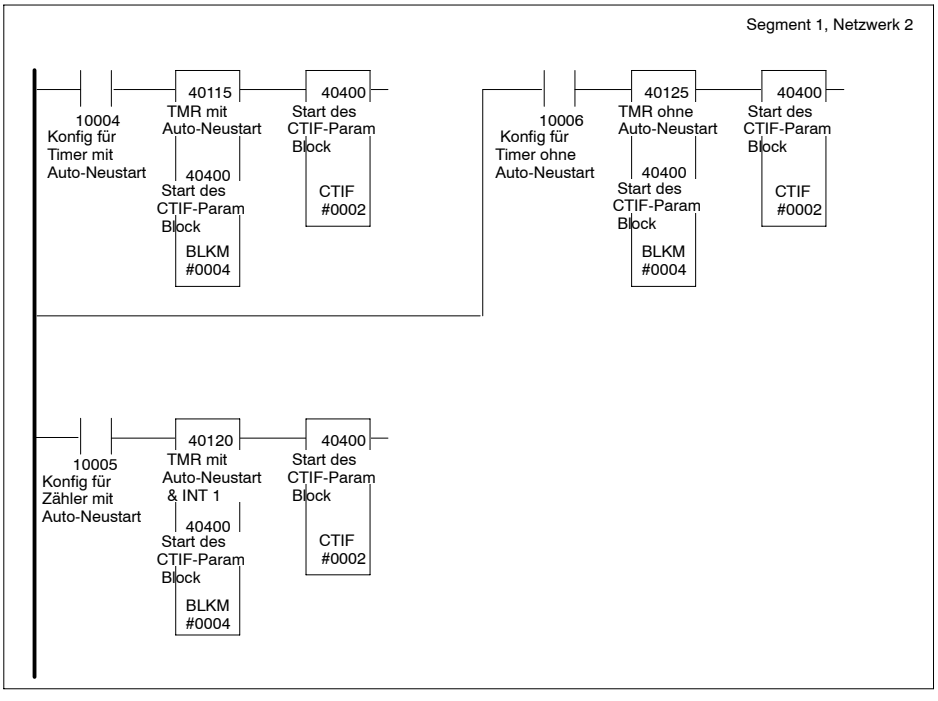
Unterprogramm 1 aufgerufen und dessen Funktion ausgeführt, d.h. der Inhalt des Registers 40501 wird um 1 erhöht.

An dieser Stelle wird die automatische Neustart-Option auf disabled gesetzt. Der Timer stellt sich auf 0 zurück, fängt jedoch erst wieder zu laufen an,wenn Kontakt 10003 von AUS zu EIN wechselt, wodurch der gesamte Ablauf von vorne beginnt. Interrupt 3 zählt die Anzahl der AUS-EIN Pegelwechsel des Eingangs.

Bei jedem Pegelwechsel des angeschlossenen Interrupt 1 wird Unterprogramm 2 aufgerufen und seine Funktion ausgeführt, d.h. der Inhalt des Registers 40502 wird um 1 erhöht.

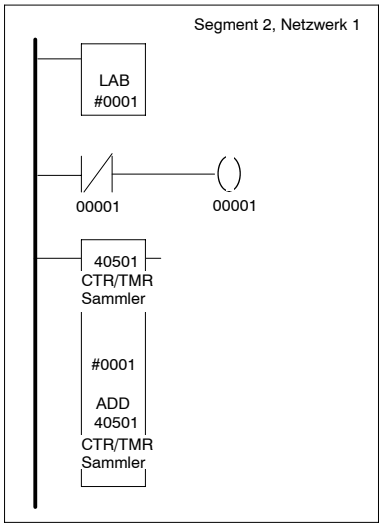
Segment 1, Netzwerk 2

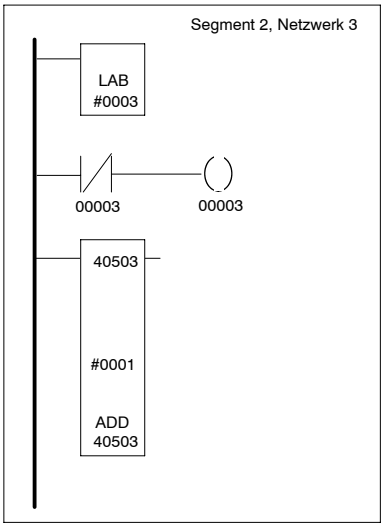
Das zweite Netzwerk in Segment 1 weist die gleiche Konfiguration auf wie das erste. Der wichtigste Unterschied ist, daß Netzwerk 2 für die Konfiguration des CTIF eines Child-SPS benutzt wird. Information von diesem Child steht der Parent-SPS nicht ohne weiteres zur Verfügung.

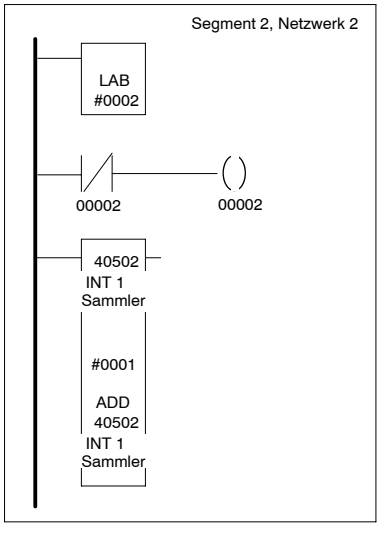


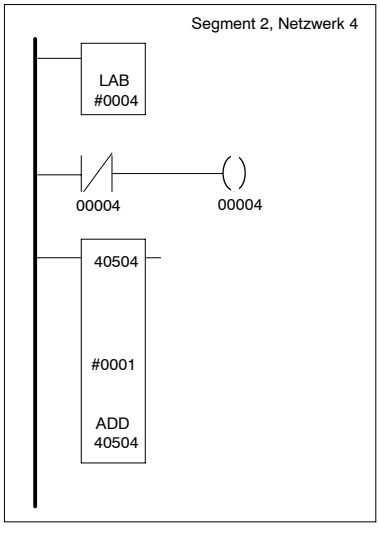
Die Unterprogramme von Segment 2

Nachfolgend sind vier Netzwerke mit Unterprogrammen aufgeführt, die durch die angeschlossenen Eingänge der vorherigen beiden Netzwerke aufgerufen werden.









Kapitel 11
Andere Standard-Befehle

- ☐Das Überspringen von Netzwerken
- ☐Prüfung des Status der SPS
- ☐Zykluszeit-Funktionen

Das Überspringen von Netzwerken

Der SKP-Befehl ermöglicht Ihnen, eine bestimmte Anzahl von Netzwerken in einem Kontaktplanlogik-Programm zu überspringen.

Wenn die SKP-Operation aktiviert ist, wird sie bei jedem Zyklusdurchlauf ausgeführt. Der Rest des Netzwerks, in dem der Befehl steht, zählt als das erste der angebenen Anzahl zu überspringender Netzwerke. Die CPU überspringt solange Netzwerke, bis die Gesamtzahl der übersprungenen Netzwerke der im Befehlsblock angegebenen Zahl entspricht oder eine Segmentgrenze erreicht wird. Eine SKP-Operation kann keine Segmentgrenze überspringen.

Ein SKP-Befehl kann nur aktiviert werden, wenn im SPS-Set-up-Editor angegeben wurde, daß Sprünge erlaubt sind.

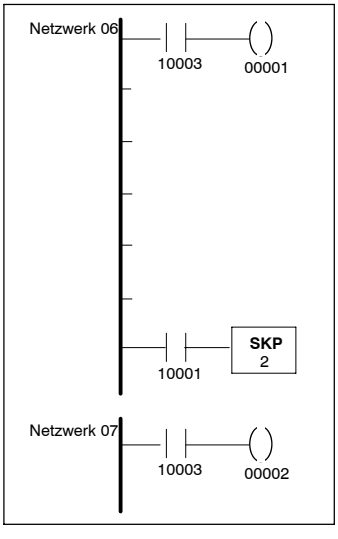
Warnung: Wenn Eingänge und Ausgänge, die normalerweise die Steuerung beeinflussen, unabsichtlich übersprungen (oder nicht übersprungen) werden, kann das Ergebnis zu Gefahren für Bedienungspersonal und Anwendungsgeräten führen.

SKP ist ein Einerelement-Befehl.

Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Überspringen von log. Netzwerken	E — SKP 3x, 4x, *oder* K*	Oben: EIN aktiviert die SKP-Funktion	Oben: Spezifiziert die Anzahl der zu über-springenden log. Netzwerke		Umgeht im Programm Netzwerke der Kontaktplanlogik und löst nicht übersprungene Logik
*K ist eine Integer-Konstante im Bereich 1 ... 255					

Ein einfaches SKP-Beispiel

Wenn Kontakt 1001 geschlossen wird, werden der Rest von Netzwerk 06 und das gesamte Netzwerk 07 übersprungen. Der Stromfluß in den übersprungenen Netzwerken ist daher ungültig. Spule 00001 wird immer noch von Kontakt 10003 gesteuert, da sie von der Logik vor dem SKP-Befehl bearbeitet wird.



Prüfung des Status der SPS

Die Modicon Micro SPS verwalten im Speicher eine Tabelle, die die lebenswichtigen Systemdiagnose-Informationen der SPS, ihrer E/A und der Kommunikation beinhalten. Diese Tabelle ist 56 Worte lang, ihr Inhalt ist wie folgt strukturiert:

Status-Wort	Inhalt des Status-Registers
1 ... 11	SPS-Statusinformation
12 ... 31	Zustand der E/A-Orte
32	Fehlercodes, die beim Systemstart erzeugt wurden
33 ... 36	Globaler Kommunikations-Status
37 ... 40	Zustand der E/A-Kommunikation der lokalen Station
41 ... 56	Zustand der E/A-Kommunikation zu und von den dezentralen Stationen

Jedes Statuswort ist 16 Bits lang und die Statusinformation wird durch den Inhalt der Bits jedes Wortes repräsentiert. Die Abbildungen auf den folgenden Seiten zeigen, wie die Statusinformation in der Statustabelle dargestellt wird.

Auf einige oder alle der Worte in der Statustabelle kann in der Kontaktplanlogik mit dem STAT-Befehl zugegriffen werden. Der STAT-Block zeigt den Inhalt der Bitmuster der Statusworte in einer Tabelle bestehend aus aufeinanderfolgenden 4 *x* Registern, deren Werte dann in der Programmiergeräte-Software betrachtet werden.

☞ **Hinweis:** Obwohl es erlaubt ist, im obersten Element entweder ein 0*x* oder 4*x* Register zu spezifizieren, empfehlen wir Ihnen, ein 4*x* Register zu benutzen, da für die Verwaltung der Statusinformation sonst eine sehr große Anzahl von 0*x* Registern erforderlich wäre.

Das Register, das Sie im obersten Block des Elements angegeben haben, wird mit den aktuellen Bitwerten von *Wort 1* geladen, darüberhinaus werden so viele Register mit Bitwerten der entsprechenden Worte der Statustabelle geladen, wie Sie im unteren Element festgelegt haben.

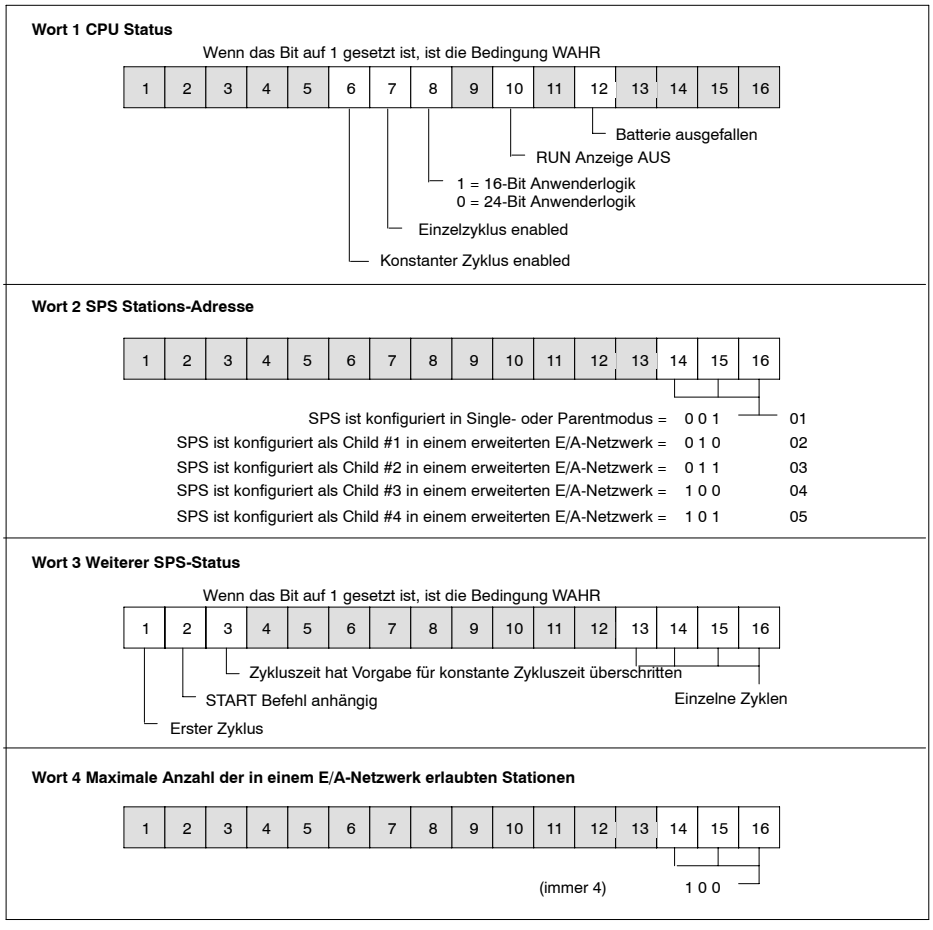
Wenn Sie z.B. nur an einem Zugriff auf die SPS-Statusinformation Interesse haben, könnten Sie eine Registeradresse wie 40701 im obersten Element des Blocks angeben und im unteren Element den Wert 11 — entsprechend werden die Bitwerte der ersten 11 Worte der Statustabelle in die Register 49701 ... 40711 geladen werden.

Wenn Sie die gesamte Statustabelle laden wollen, müssen Sie im unteren Element 56 einsetzen. Falls Sie keine E/A-Erweiterung benutzen, reicht es, im unteren Element den Wert 40 einzugeben, um die relevante Statusinformation zu erhalten.

STAT ist ein Zweierelement-Befehl.

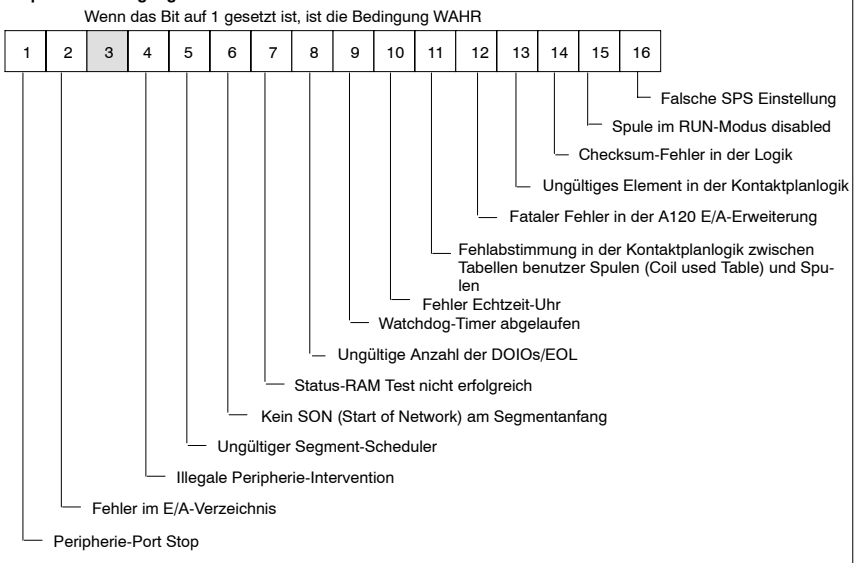
Befehl	Struktur	Eingänge (E)	Element	Ausgänge (A)	Funktion
Prüfung CPU/ E/A-Status	E — 0*x* oder 4*x* — A STAT K*	Oben: EIN Zugriff auf die Statustabelle	Oben: Erstes Wort in der System-Statustabelle Unten: Länge der Statustabelle	Oben: Operation ausgeführt	Holt Statusdaten aus der Status- tabelle im Systemspeicher und zeigt diese in Anwenderregistern
*K ist eine Integer-Konstante im Bereich 1 ... 56					

Die Statustabelle der Modicon Micro SPS

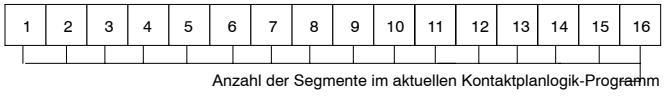

Wort 1 CPU Status
Wenn das Bit auf 1 gesetzt ist, ist die Bedingung WAHR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Batterie ausgefallen
RUN Anzeige AUS
1 = 16-Bit Anwenderlogik
0 = 24-Bit Anwenderlogik
Einzelzyklus enabled
Konstanter Zyklus enabled
Wort 2 SPS Stations-Adresse
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
SPS ist konfiguriert in Single- oder Parentmodus = 0 0 1 01
SPS ist konfiguriert als Child #1 in einem erweiterten E/A-Netzwerk = 0 1 0 02
SPS ist konfiguriert als Child #2 in einem erweiterten E/A-Netzwerk = 0 1 1 03
SPS ist konfiguriert als Child #3 in einem erweiterten E/A-Netzwerk = 1 0 0 04
SPS ist konfiguriert als Child #4 in einem erweiterten E/A-Netzwerk = 1 0 1 05
Wort 3 Weiterer SPS-Status
Wenn das Bit auf 1 gesetzt ist, ist die Bedingung WAHR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Zykluszeit hat Vorgabe für konstante Zykluszeit überschritten
START Befehl anhängig
Einzelne Zyklen
Erster Zyklus
Wort 4 Maximale Anzahl der in einem E/A-Netzwerk erlaubten Stationen
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(immer 4) 1 0 0

Statustabelle der Modicon Micro SPS (Fortsetzung)

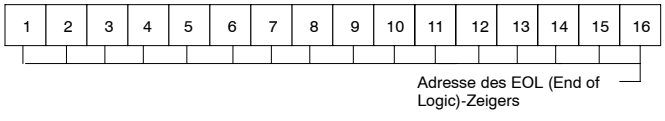
Wort 5 CPU Stopstatus-Bedingungen


Wenn das Bit auf 1 gesetzt ist, ist die Bedingung WAHR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Falsche SPS Einstellung
Spule im RUN-Modus disabled
Checksum-Fehler in der Logik
Ungültiges Element in der Kontaktplanlogik
Fataler Fehler in der A120 E/A-Erweiterung
Fehlabstimmung in der Kontaktplanlogik zwischen Tabellen benutzer Spulen (Coil used Table) und Spulen
Fehler Echtzeit-Uhr
Watchdog-Timer abgelaufen
Ungültige Anzahl der DOIOs/EOL
Status-RAM Test nicht erfolgreich
Kein SON (Start of Network) am Segmentanfang
Ungültiger Segment-Scheduler
Illegale Peripherie-Intervention
Fehler im E/A-Verzeichnis
Peripherie-Port Stop

Wort 6 Segmente im Programm



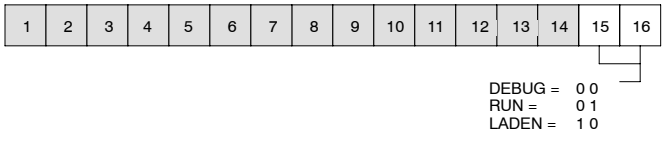
Wort 7 Logikende-Zeiger



Wort 8 reserviert

Wort 9 reserviert

Wort 10 RUN/LOAD/DEBUG Status



Wort 11 reserviert

Statustabelle der Modicon Micro SPS (Fortsetzung)

Worte 12 ... 31 Zustand der E/A-Einheiten

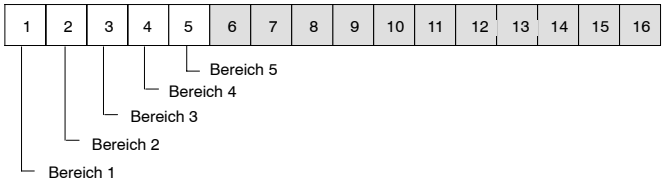
Vier aufeinanderfolgende Worte werde für jede von bis zu 5 SPS in einem E/A-Erweiterungsnetz genutzt. Ein Wort in jeder Vierergruppe wird für jeden möglichen E/A-Baugruppenträger belegt, in der Annahme, daß es sich um A120 Erweiterung handelt:

Wort	SPS	Baugruppenträger
12	1	1
13		2
14		3
15		4
16	2	1
17		2
18		3
19		4
20	3	1
21		2
22		3
23		4
24	4	1
25		2
26		3
27		4
28	5	1
29		2
30		3
31		4

Baugruppenträger 1 ist immer eine Modicon Micro SPS, und Baugruppenträger 2 ... 4 sind A120 E/A, angeschlossen an Baugruppenträger 1 über einen A120 E/A-Erweiterungsport.

Jedes Wort enthält fünf repräsentative Bits, die den Zustand der assoziierten E/A-Einheit jedes Baugruppentägers zeigen — d.h. jeder Baugruppenträger kann ein Maximum von fünf E/A-Bereichen unterstützen:

Wenn das Bit auf 1 gesetzt ist, ist die E/A-Einheit im Baugruppenträger in Ordnung



Bezüglich der A120 E/A Module ist ein Ort die physikalische Steckplatz-Position des Moduls in seinem DTA-Gehäuse. Bezüglich einer Modicon Micro *SPS* bezieht sich der Ort auf die folgenden, festen Komponenten auf der Einheit:

- Bereich 1 repräsentiert die festen, diskreten Eingänge und Ausgänge auf der Einheit
- Bereich 2 repräsentiertt den Komponentenstatus des dezidierten Interrupts auf der Einheit
- Bereich 3 repräsentiert den vom Anwender wählbaren Zähler/Timer Stand auf der Einheit
- Bereich 4 repräsentiert beliebige feste, analogeEingänge und Ausgänge auf der Einheit
- Bereich 5 repräsentiert die Datentransfer-Komponente für serielle E/A-Erweiterung auf der Einheit

Ein E/A-Bereich ist in Ordnung, wenn er konfiguriert und korrekt in das E/A-Verzeichnis aufgenommen wurde, seine Identität richtig ist und zwischen ihm und der ihn steuernden CPU eine gültige Kommunikation besteht.

Konvertierung von Wort # zu SPS und Baugruppenträger

$$\frac{Wort\ \#\ -\ 12}{4} = Quotient + Rest$$

wobei

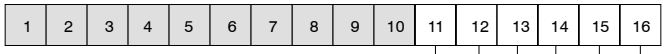
Quotient + 1 = Station #
Rest + 1 = Baugruppenträger #

Konvertierung von Station und Baugruppenträger zu Wort #

Wort # = (*Station #* x 4) + *Baugruppenträger #* + 7

Statustabelle der Modicon Micro SPS (Fortsetzung)

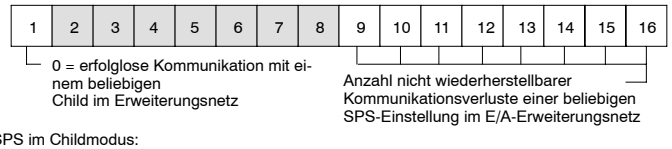
Wort 32 Start-Fehlercodes (immer 0 , wenn System ordnungsgemäß läuft)



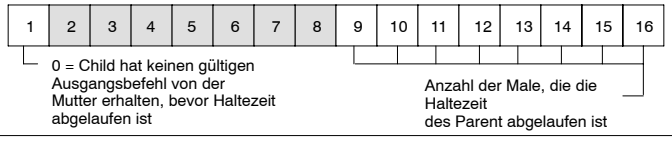
Falsche E/A-Verzeichnislänge	0 0 0 0 0 1	01
Falsche Kopplungsnummer für Child-SPS im Netzwerk	0 0 0 0 1 0	02
Falsche Nummer für Child-SPS im E/A-Verzeichnis	0 0 0 0 1 1	03
Falsche Prüfsumme E/A-Verzeichnis	0 0 0 1 0 0	04
Falsche Kommentarlänge bei Child-SPS	0 0 1 0 1 0	10
Falsche Nummer der Child-SPS	0 0 1 0 1 1	11
Falsche Haltezeit für Child-SPS im Netzwerk	0 0 1 1 0 0	12
Falsche ASCII Portnummer	0 0 1 1 0 1	13
Falsche Anzahl von Steckplätzen in einer Child-SPS	0 0 1 1 1 0	14
Child-SPS wurde schon eingerichtet	0 0 1 1 1 1	15
Comm Port wurde schon eingerichtet	0 1 0 0 0 0	16
Mehr als 1024 Ausgangspunkte	0 1 0 0 0 1	17
Mehr als 1024 Eingangspunkte	0 1 0 0 1 0	18
Falsche Steckplatzadresse	0 1 0 1 0 0	20
Falsche Baugruppenträger-Adresse	0 1 0 1 0 1	21
Falsche Anzahl von Ausgangs-Bytes	0 1 0 1 1 0	22
Falsche Anzahl von Eingangs-Bytes	0 1 0 1 1 1	23
Erste Referenznummer falsch	0 1 1 0 0 1	25
Zweite Referenznummer falsch	0 1 1 0 1 0	26
Keine Eingangs- oder Ausgangs-Bytes	0 1 1 0 1 1	27
Diskrete nicht auf einer 16-Bit Grenze	0 1 1 1 0 0	28
Nicht-paarige, ungerade Ausgangseinheit	0 1 1 1 1 0	30
Nicht-paarige, ungerade Ausgangseinheit	0 1 1 1 1 1	31
Nicht passende gerade Eingangs-/Ausgangs-Einheitenreferenz	1 0 0 0 0 0	32
1*x* Referenz nach 3*x* Registern	1 0 0 0 0 1	33
Referenz für Dummyeinheit wird bereits benutzt	1 0 0 0 1 0	34
3*x* Referenz kein Dummy	1 0 0 0 1 1	35
4*x* Referenz kein Dummy	1 0 0 1 0 0	36
Dummy, danach richtige 1*x* Referenz	1 0 1 0 0 0	40
Richtige, danach Dummy 1*x* Referenz	1 0 1 0 0 1	41
Dummy, danach richtige 3*x* Referenz	1 0 1 0 1 0	42
Richtige, danach Dummy 3*x* Referenz	1 0 1 0 1 1	43
Zu viele E/A-Punkte in einer Station	1 0 1 1 0 0	44
Falsche Einheitenbeschreibung Baugruppenträger	1 1 0 0 1 0	50
Falsche Einheitenbeschreibung Steckplatz	1 1 0 0 1 1	51
Falsche Einheitenbeschreibung Eingangsbyte-Zählung	1 1 0 1 0 0	52
Falsche Einheitenbeschreibung Ausgangsbyte-Zählung	1 1 0 1 0 1	53
E/A-Treiber nicht geladen	1 1 0 1 1 0	54
Einheit kann nur in Baugruppenträger 1 benutzt werden	1 1 0 1 1 1	55

Wort 33 Globale Kommunikation

für eine SPS im Parent-oder Singlemodus:


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 = erfolglose Kommunikation mit einem beliebigen Child im Erweiterungsnetz
Anzahl nicht wiederherstellbarer Kommunikationsverluste einer beliebigen SPS-Einstellung im E/A-Erweiterungsnetz

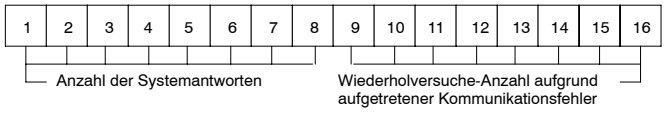
für eine SPS im Childmodus:


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 = Child hat keinen gültigen Ausgangsbefehl von der Mutter erhalten, bevor Haltezeit abgelaufen ist
Anzahl der Male, die die Haltezeit des Parent abgelaufen ist

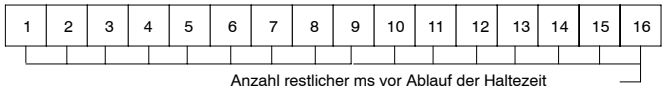
Statustabelle der Modicon Micro SPS (Fortsetzung)

Wort 34 Zusätzliche globale Kommunikation

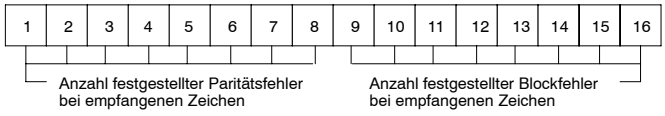
für eine SPS im Modus Parent:


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Anzahl der Systemantworten
Wiederholversuche-Anzahl aufgrund aufgetretener Kommunikationsfehler

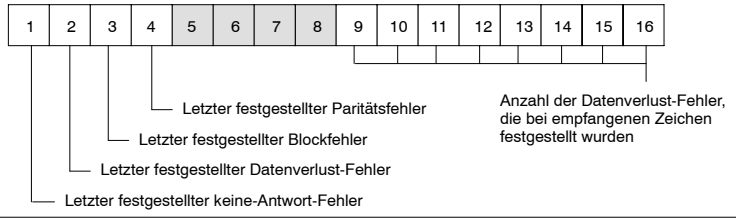
für eine SPS im Modus Child:



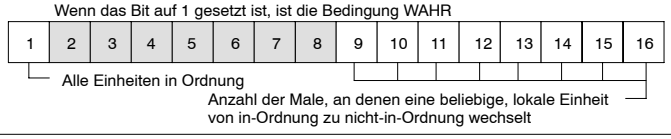
Wort 35 Zusätzliche globale Kommunikation
(nur für eine SPS im Modus Parent)


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Anzahl festgestellter Paritätsfehler bei empfangenen Zeichen
Anzahl festgestellter Blockfehler bei empfangenen Zeichen

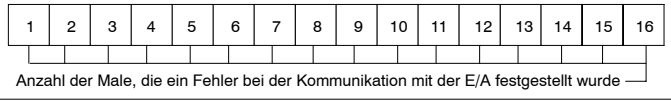
Wort 36 Zusätzliche, globale Kommunikation
(nur für eine SPS im Modus Parent)


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Letzter festgestellter Paritätsfehler
Anzahl der Datenverlust-Fehler, die bei empfangenen Zeichen festgestellt wurden
Letzter festgestellter Blockfehler
Letzter festgestellter Datenverlust-Fehler
Letzter festgestellter keine-Antwort-Fehler

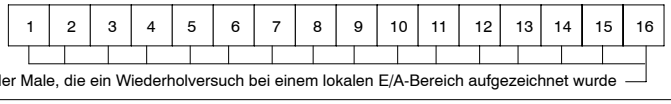
Worte 37 Ordnungsgemäße Kommunikation in Baugruppenträger 1
(nur bei A120 Erweiterung)


Wenn das Bit auf 1 gesetzt ist, ist die Bedingung WAHR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Alle Einheiten in Ordnung
Anzahl der Male, an denen eine beliebige, lokale Einheit von in-Ordnung zu nicht-in-Ordnung wechselt

Worte 38 E/A-Fehlerermittlung in Baugruppenträger 1
(nur bei A120 E/A-Erweiterung)



Worte 39 E/A Retry-Zähler in Baugruppenträger 1
(nur bei A120 E/A-Erweiterung)


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Anzahl der Male, die ein Wiederholversuch bei einem lokalen E/A-Bereich aufgezeichnet wurde

Wort 40 reserviert, alle Bits sind 0

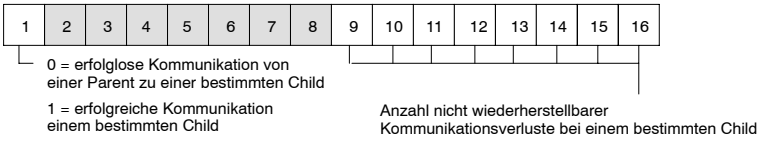
Statustabelle der Modicon Micro SPS (Fortsetzung)

Worte 41 ... 56 sind für Kommunikation in dem E/A-Erweiterungsnetzwerk= sie haben nur für die Parenteinheiten Bedeutung

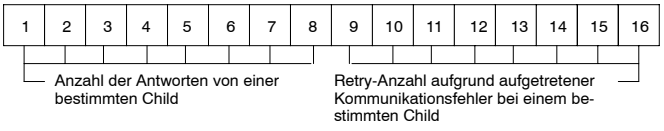
Jedes potentielle Child im Netzwerk wird durch eine Gruppe von vier aufeinanderfolgenden Worten beschrieben:

- Worte 41 ... 44 gehören zu Child #1
- Worte 45 ... 48 gehören zu Child #2
- Worte 49 ... 52 gehören zu Child #3
- Worte 53 ... 56 gehören zu Child #4

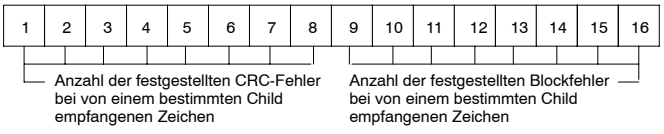
Wort 41, 45, 49, 53 Format


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 = erfolglose Kommunikation von einer Parent zu einer bestimmten Child
1 = erfolgreiche Kommunikation einem bestimmten Child
Anzahl nicht wiederherstellbarer Kommunikationsverluste bei einem bestimmten Child

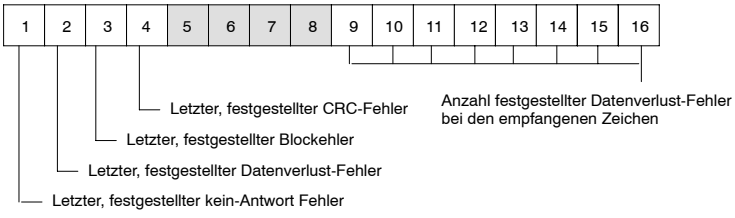
Wort 42, 46, 50, 54 Format


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Anzahl der Antworten von einer bestimmten Child
Retry-Anzahl aufgrund aufgetretener Kommunikationsfehler bei einem bestimmten Child

Wort 43, 47, 51, 55 Format


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Anzahl der festgestellten CRC-Fehler bei von einem bestimmten Child empfangenen Zeichen
Anzahl der festgestellten Blockfehler bei von einem bestimmten Child empfangenen Zeichen

Wort 44, 48, 52, 56 Format


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Letzter, festgestellter CRC-Fehler
Anzahl festgestellter Datenverlust-Fehler bei den empfangenen Zeichen
Letzter, festgestellter Blockehler
Letzter, festgestellter Datenverlust-Fehler
Letzter, festgestellter kein-Antwort Fehler

Zykluszeit-Funktionen

Zykluszeit-Funktionen ermöglichen eine Logikbearbeitung in festen Zeitinvallen — die Logik der Steuerung kann jedoch dadurch nicht schneller abeiten oder Zyklusdurchläufe vorzeitig abbrechen. Zykluszeit-Funktionen können *konstant* oder für eine festgelegte Anzahl von Zyklusdurchläufen gelten=d.h. *einzelne Zyklen* sein.

Mit einem *konstanten Zyklus* können die Zielzeiten der Zyklusdurchläufe von 10 ... 200 ms (Mehrfache von 10 ms) betragen. Die Ziel-Zykluszeit ist die Zeit, die zwischen dem Beginn eines Zyklusses und dem Beginn des nächsten Zyklus liegt. Wenn ein konstanter Zyklus mit einem Zeitdurchlauf eingerichtet wird, der kürzer ist als die eigentlich Zykluszeit, wird diese Zykluszeit-Funktion ignoriert und das System benutzt seine normale Zyklusdurchlaufzeit.

Eine längere Durchlaufzeit bei einem konstanten Zyklus wirkt sich auf die Logikbearbeitungszeit, E/A und Modbus Port-Verwaltung sowie die Systemdiagnose aus. Mit einem konstanten Zyklus können die Zielzeiten der Zyklusdurchläufe von 10 ... 200 ms (Mehrfache von 10 ms) betragen. Wenn Sie eine konstante Ziel-Zykluszeit von 40 ms festlegen und die eigentliche Logikbearbeitung, Port-Verwaltung und Diagnose nur 30 ms benötigt, wird die Steuerung bei jedem Zyklusdurchlauf 10 ms warten.

Einzel-Zyklus Funktionen ermöglicht Ihrer Steuerung, eine festgelegte Anzahl an Zyklusdurchläufen auszuführen — von 1 ... 15 — und dann die Logikbearbeitung anzuhalten und die Ein-/Ausgabe zu bedienen. Diese Funktion ist für Diagnosezwecke hilfreich. Sie erlaubt die Fehlerprüfung von bearbeiteter Logik, verschobenen Daten und ausgeführten Berechnungen.

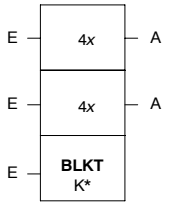
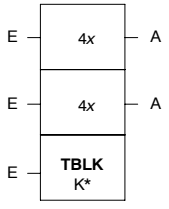
Warnung: Einzel-Zyklen sollten nicht für die Fehlersuche bei Steuerungen von Maschinenwerkzeugen, Prozessen oder Materialverwaltungssystemen eingesetzt werden, wenn diese laufen. Wenn einmal die angegebene Anzahl von Zyklusdurchläufen abgearbeitet wurde, werden alle Ausgänge in ihrem letzten Zustand konserviert. Da keine Logikbearbeitung mehr stattfindet, ignoriert die Steuerung alle Eingangsinformationen. Dies kann zu unsicheren, gefährlichen und destruktiven Operationen der Werkzeuge oder Prozesse, die an der Steuerung angeschlossen sind, führen.

In Ihrer Programmier-Dokumentation finden Sie weitere Informationen über den Gebrauch von Zykluszeit-Funktionen.

Kapitel 12
Erweiterter Befehlssatz verfügbar auf ausgewählten Micro SPS Modellen

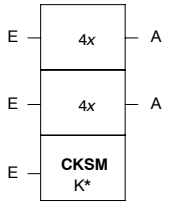
- ☐Block↔Tabelle Verschiebungsbefehle
- ☐Der Checksum-Befehl
- ☐Der Proportional-Integral-Differerential-(PID)-Befehl
- ☐Erweiterte mathematische Befehle

Block↔Tabelle Verschiebungsbefehle

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Block-zu-Tabelle Verschiebung		Oben: EIN initiiert die Verschiebung Mitte: EIN hält den Zeiger an Unten: EIN stellt den Zeiger auf 0 zurück	Oben: Erstes Register im Quellblock Mitte: Zeiger zum ersten Register (4x + 1) in der Zieltabelle Unten: Länge derZieltabelle	Oben: EIN, wenn Operation beendet Mitte: Fehler festgestellt—Verschiebung nicht möglich	Verschiebt große Mengen an 4*x* Registern von einem festen Quellblock zu einem Ziel in einer Tabelle
Tabelle-zu-Block Verschiebung		Oben: EIN initiiert die Verschiebung Mitte: EIN hält den Zeiger an Unten: EIN setzt den Zeiger auf 0 zurück	Oben: Erstes Register in der Quelltabelle Mitte: Zeiger auf das erste Register (4x + 1) im Zielblock Unten: Länge des Zielblocks	Oben: EIN, wenn Operation beendet Mitte: Fehler festgestellt—Verschiebung nicht möglich	Verschiebt große Mengen aufeinanderfolgender Register in einer Tabelle zu einem festen Zielblock

*K ist eine Integer-Konstante im Bereich 1 ... 100

Der Checksum-Befehl

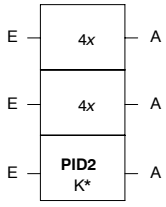
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Checksum		Oben: EIN berechnet die Quelltabellen cksm Mitte: Benutzt mit dem unteren Eingang, um cksm-Typ festzustellen Unten: Benutzt mit dem mittleren Eingang, um cksm-Typ festzustellen	Oben: Erstes Register in der Quelltabelle Mitte: Erstes von zwei Registern, die das Ergebnis und die implizite Registerzählung enthalten Unten: Länge der Quelltabelle	Oben: EIN, wenn Berechnung beendet Mitte: Fehler festgestellt: Registerzählung = 0 oder Registerzählung > Größe der Quelltabelle	Führt lineare Prüfung aus, binäre Additionsprüfung, CRC-16 Prüfung oder LRC Prüfung, abhängig vom Status des mittleren und unteren Eingangs (vgl. Tabelle unten)

*K ist eine Integer-Konstante im Bereich 1 ... 255

CKSM Eingangsbenutzung

CKSM Berechnung	Mitte Eingang	Unten Eingang
Lineare Prüfung	AUS	EIN
Binäre Addition	EIN	EIN
CRC-16	EIN	AUS
LRC	AUS	AUS

Der Proportional-Integral-Differerential-Befehl

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Proportional-Integral-Differential		Oben: 0 = Manuller Modus 1 = Auto Modus Mitte: 0 = Aufzeichnung EIN 1 = Aufzeichnung AUS Unten: 0 = Ausgangswert wird höher mit kleiner werdendem E** 1 = Ausgangswert wird kleiner mit größer werdendem E**	Oben: Erstes von 21 Registern in der Quelltabelle Mitte: Erstes von 9 Registern, das vom Block für Berechnungen benutzt wird Unten: Konstante, die das Intervall repräsentiert, in dem die Berechnung in Zehntelsekun den ausgeführt wird	Oben: ungültiger Parameter oder aktive Programmschleife wird nicht gelöst Mitte: PV ≥ untere Alarmgrenze*** Unten: PV ≥ untere Alarmgrenze***	Implementiert einen Algorithmus, der die spezifizierteP, PI, oder PID Operation,wie in den Registern 4*x* + *5*, 4*x* + *6*, 4*x* + *7 und* 4x + 8 der Quelltabelle definiert, ausführt

* K ist eine Integer-Konstante im Bereich 1 ... 255

** E ist ein Fehler, der in Roh-Analogeinheiten dargestellt wird

*** PV ist die Prozeßvariable

Block-funktion	Registerwert Quelltabelle			
	4*x* + 5	**4*x* + 6**	**4*x* + 7**	**4*x* + 8**
P	ungleich Null	Null	Null	ungleich Null
PI	ungleich Null	ungleich Null	Null	Null
PI	ungleich Null	ungleich Null	ungleich Null	Null

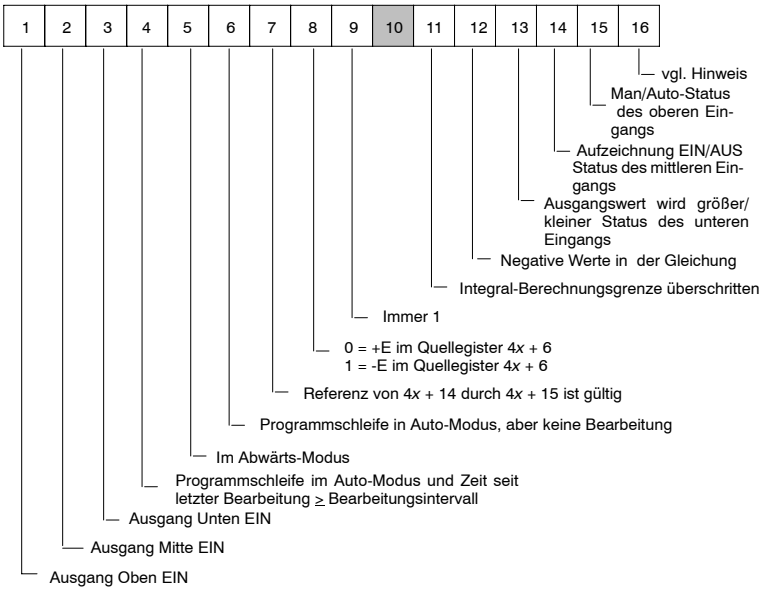
PID2 Quelltabelle (oberers Element)

Registernummerr	Registerinhalt
4*x*	**Skalierte PV**: bei jedem Zyklusdurchlauf vom Block geladen. Eine lineare Skalierung wird in Register 4*x* + 13 durchgeführt unter Gebrauch der hohen und niedrigen Bereiche in 4*x* + 11 und 4*x* +12: $\text{skalierte PV} = \frac{\text{reg } 4x + 13}{4095} \times (\text{reg } 4x + 11 - \text{reg } 4x + 12) + \text{reg } 4x + 12$ Der Rest wird am Dezimalpunkt abgeschnitten und alle Ziffern rechts des Dezimalpunkts gelöscht—ohne Abrundung des Ergebnisses.
*4*x + 1	**SP**: Der Sollwert wird in technischen Einheiten spezifiziert, deren Wert sein muß > 4*x* + 11 > 4*x* + 12
*4*x + 2	**M_V:** bei jedem Zyklusdurchlauf vom Block geladen, gekoppelt an den Bereich 0 ... 4095, wodurch der Ausgang zu einem analogen Ausgang kompatibel wird. Die manipulierte Registervariable kann für weitere CPU-Berechnungen wie Kaskaden-Programmschleifen benutzt werden
*4*x + 3	**Obere Alarmgrenze**: Laden eines Wertes in dieses Register, um für die PV einen oberen Alarm anzugeben (gleich oder über dem SP). Geben Sie den Wert in technischen Einheiten innerhalb des in den Registern4*x* + 11 und 4*x* + 12 spezifizierten Bereichs ein
*4*x + 4	**Untere Alarmgrenze:** Laden eines Wertes in dieses Register, um für die PV einen unteren Alarm anzugeben (gleich oder unter dem SP). Geben Sie den Wert in technischen Einheiten innerhalb des in den Registern 4*x* + 11 und 4*x* + 12 spezifizierten Bereichs ein
*4*x + 5	**Proportionalbereich:** Laden Sie dieses Register mit der gewünschten proportionalen Konstante im Bereich 5 ... 500. Je kleiner die Zahl, desto größer der Proportionlanteil. In diesem Register ist eine gültige Zahl erforderlich,damit PID2 arbeiten kann

Quelltabelle (oberer Elemente)	
Register-nummer	**Registerinhalt**
4x + 6	**Integralzeit-Konstante:** Laden Sie dieses Register, um der Berechnung Intergralverhalten hinzuzufügen. Der Wert ist eine Integer-Konstante im Bereich 0000 ... 9999, die einen Bereich von 00.00 ... 99.99 Wiederholungen pro Minute darstellt—Werte <9999 oder >0000 stoppen die PID2-Berechnung. Je größer die Zahl, dest größer der Integralanteil
4x + 7	**Differential-Zeitkonstante:** Laden Sie dieses Register, um der Berechnung Differentialverhalten hinzuzufügen. Der Wert ist eine Integer-Konstante im Bereich 0000 ... 9999, die einen Bereich von of 00.00 ... 99.99 Wiederholungenpro Minute darstellt—Werte <9999 oder >0000 stoppen die PID2-Berechnung. Je größer die Zahl, dest größer der Differentialanteil
4x + 8	**Vorspannung:** Laden Sie dieses Register, um dem Ausgang eine Vorspannung hinzuzufügen=der Wert, der direkt zu $M_v$addiert wird, muß zwischen 0000 ... 4095 liegen
4x + 9	**Obere Integral Berechnungsgrenze:** Laden Sie dieses Register mit der oberen Grenze des Ausgangswerts (zwischen 0 ... 4095), an der die Anti-Integral Berechnung stattfindet. Wenn der spezifizierte Wert (normalerweise 4095) überschritten wird, wird die Integralsumme nicht mehr aktualisiert
4x + 10	**Untere Integral Berechnungsgrenze:** Laden Sie dieses Register mit der unteren Grenze des Ausgangswerts (zwischen 0 ... 4095), an der die Anti-Integral Berechnung stattfindet—der angegebene Wert ist normalerweise 0
4x + 11	**Obergrenze technischer Bereich:** Laden Sie dieses Register mit dem höchsten Wert, den das Meßgerät umfaßt—d.h.., wenn ein Widerstands-Temperaturgerät von 0 ... 500 Grad C reicht, ist der höchste technische Bereichswert 500; der höchste Bereichswert muß als positive Integerzahl zwischen 0001 ... 9999 angegeben werden, entsprechend den Roh-Analogeingangswerten des 4095
4x + 12	**Untergrenze technischer Bereich:** Laden Sie dieses Register mit dem niedrigsten Wert, den das Meßgerät umfaßt. Der untere Bereichswert muß als positive Integerzahl zwischen 0001 ... 9998 angegeben werden, entsprechend einem Roh-Analogeingangswert von 0—er muß kleiner sein als der in Register 4x + 11 angegebene Wert
4x + 13	**Roh-Analogmessung:** das Logikprogramm lädt dieses Register mit der PV. Die Messung muß skaliert sein und linear im Bereich 0 ... 4095
4x + 14	**Zeiger auf Programmschleifen Register:** Der Wert, den Sie in dieses Register laden, verweist auf das Register, das die Anzahl der Programmschleifen in jedem Zyklusdurchlauf zählt. Der in das Register einzugebende Wert ist die Referenz-nummer des Registers, das die Programmschleife enthält—wenn z.B. Register 41236 den Zählerwert enthält, müssen Sie den Wert 1236 in Register 4x + 14 der PID2 Quelltabelle eingeben; der gleiche Wert muß in die 4x + 14 Register in der Quelltabelle jedes PID2 Block in einem Logikprogramm geladen werden
4x + 15	**Maximale Anzahl der Programmschleifen/Zyklus:** Wenn Register 4x = 14 einen Wert ungleich Null enthält, können Sie einen Wert in dieses Register laden, um die obere Grenze der Programmschleifen anzugeben, die in einem einzelnen Zyklusdurchlauf durchgeführt werden sollen
4x + 16	**Zeiger zum Rücksetzen des Rückkopplungseingangs:** der Wert, den Sie in dieses Register laden, zeigt auf das Halteregister, das den Rückkopplungswert enthält (F). Integrationsberechnungen hängen von dem F Wert ab, der mitM_v verbunden ist—da der PID2 Ausgang von0 ... 4095 variiert, sollte F ebenfalls von 0 ... 4095 variieren. Der in das Register eingegebne Wert ist die Referenznummer des Rückkopplungsregisters—wenn z.B. das Rückkopplungsregister 42250 ist, müssen Sie den Wert 2250 in Register 4x + 16 der PID2 Quelltabelle eingeben
4x + 17	**Clampingausgang Obergrenze:** der in dieses Register eingebene Wert legt die obere Grenze von M_v fest (normalerweise 4095)
4x + 18	**Clampingausgang Untergrenze:** der in dieses Register eingegene Wert legt die untere Grenze von M_v fest (normalerweise 0)
4x + 19	**RGL-Konstante:** der in dieses Register eingegebene Differentialbegrenzungswert bestimmt den Effektivgrad der Differentialdämpfung. Der Bereich für diesen Wert ist 2 ... 30. Je kleiner der Wert, desto höher die Dämpfung.
4x + 20	**Zeiger auf Aufzeichnungseingang:** Der in dieses Register eingegebene Wert zeigt auf das Halteregister, das den Wert des Aufzeichnungseingangs (T) enthält. Der Wert T wird an den Eingang der Integralverzögerung gekoppelt, sobald das Auto-Bit und Aufzeichnungs-Bit beide WAHR sind. Der in dieses Register eingegebene Wert ist die Register-Referenznummer des Aufzeichnungseingangs—wenn z.B. 40956 das Aufzeichnungseingangs-Register ist, müssen Sie 0956 in Register 4x + 20 in der PID2 Quelltabelle eingeben

PID2 Berechnungsblock (mittleres Element)

Registernummer	Registerinhalt
4x	Programmschleifenstatus-Register



Hinweis: Bit 16 wird nach dem ersten Starten oder Einrichten der Schleife gesetzt. Ist das Bit gelöscht, werden in einem Zyklus die folgenden Aktionen ausgeführt:

Das Programmschleifen-Statusregister wird zurückgesetzt

Der aktuelle Wert der Echtzeit-Uhr wird in Register 4*x* + 1 dieses Blocks gespeichert

Register 4*x* + 3, 4*x* + 4 und 4*x* + 5 dieses Blocks werden auf Null gesetzt

Der Wert im Quelltabellen-Register 4*x* + 13 wird mit 8 multipliziert und in Register 4*x* + 6 dieses Blocks gespeichert

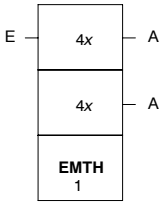
Register 4*x* + 7 und 4*x* + 8 dieses Blocks werden gelöscht

PID2 Berechnungsblock (Mittleres Element)

Registernummer	Registerinhalt		
4x + 1	Fehlerstatus (E)		
	Bit-Kode	**Bedeutung**	**Prüfen Sie dieses Register in der Quelltabelle (Oberes Element)**
	0000	Keine Fehler, alle Prüfungen OK	
	0001	Skalierter SP über 9999	4*x* + 1
	0002	Alarm Obergrenze über 9999	4*x* + 3
	0003	Alarm Untergrenze über 9999	4*x* + 4
	0004	Proportionalbereich unter 5	4*x* + 5
	0005	Proportionalbereich über 500	4*x* + 5
	0006	Rücksetzen öfter als 99.99 Wiederholungen/min	4*x* + 6
	0007	Zeitbereich über 99.99 min	4*x* + 7
	0008	Vorspannung über 4095	4*x* + 8
	0009	Integral Obergrenze über 4095	4*x* + 9
	0010	Integral Untergrenze über 4095	4*x* + 10
	0011	Obergrenze technische Einheitenskalierung über 9999	4*x* + 11
	0012	Untergrenze technische Einheitenskalierung über 9999	4*x* + 12
	0013	Obergrenze t. Ein.skal. unter Ubergrenze t. Einheiten	4*x* + 11 und 4*x* + 12
	0014	Skalierter SP über Obergrenze technische Einheiten	4*x* + 1 und 4*x* + 11
	0015	Skalierter SP unter Untergrenze technische Einheiten	4*x* + 1 und 4*x* + 11
	0016	Programmschleifen/Zyklusdurchläufe > 9999	(4*x* + 15 = 0)
	0017	Rücksetzen Rückkopllungszeigers außerhalb d. Bereichs	4*x* + 16
	0018	Oberer Clampingausgang über 4095	4*x* + 17
	0019	Unterer Clampingausgang über 4095	4*x* + 18
	0020	Unterer Clampingausgang ü. oberem Clampingausgang	4*x* + 17 und 4*x* + 18
	0021	RGL unter 2	4*x* + 19
	0022	RGL über 30	4*x* + 19
	0023	Aufzeichnung F Zeiger außerhalb des Bereichs	4*x* + 20 und mittlerer Eingang EIN
	0024	Aufzeichnung F Zeiger ist Null	4*x* + 20 und mittlerer Eingang EIN
	0025	Elemente gesperrt (zu knappe Zykluszeit)	vgl. Hinweis unten
	0026	Programmschleifen-Zählerzeiger ist Null	4*x* + 14 und 4*x* + 15
	0024	Programmschleifen-Zählerzeiger außerhalb des Bereichs	4*x* + 14 und 4*x* + 15
	Hinweis: Wenn eine Sperre häufiger auftritt und alle Parameter gültig sind, sollte die höchste, zulässige Anzahl der Programmschleifen pro Zyklusdurchlauf erhöht werden. Eine Sperre kann auch auftreten, wenn die gebrauchten Zählregister nicht wie erforderlich gelöscht werden.		
4x + 2	**Programmschleifen-Timerregister:** speichert die Echtzeit-Uhr und liest die Systemuhr bei der Bearbeitung jeder Programmschleife aus. Die Differenz zwischen dem aktuellen Uhrenwert und dem in diesem Register gespeicherten Wert ist die vergangene Zeit. Wenn die vergangene Zeit ≥ dem Bearbeitungsintervall ist (10 mal der Wert in dem unteren Elemente des PID2 Blocks), müßte die Programmschleife im aktuellen Zyklusdurchgang gelöst werden.		
4x + 3 4x + 4 4x + 5	Reserviert für internen Gebrauch		

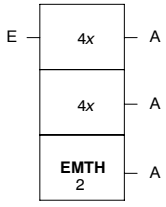
PID2 Berechnungsblock (mittleres Element)	
Registernummer	**Registerinhalt**
*4*x + 6	P_V x 8 (Dämpfung): speichert das Ergebnis des gedämpften analogen Eingangs (aus Quellregister 4*x* + 14) multipliziert mit acht. Dieser Wert ist nützlich für Differential-Steueroperationen
*4*x + 7	Absoluter Wert von E: enthält den absoluten Wert des SP - PV; Bit 8 in Register 4*x* + 1 dieses Blocks gibt das Vorzeichen von E an. Der Wert in diesem Register wird nach jeder Abarbeitung der Programmschleife aktualisiert
*4*x + 8	Reserviert für internen Gebrauch

Erweiterte mathematische Befehle

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Doppelte Genauigkeit (32-Bit) Addition		Oben: EIN initiiert die Addition mit doppelter Genauigkeit	Oben: Erstes von zwei aufeinanderfolgenden Registern, das Operand 1 enthält—sein Wert liegt im Bereich 0 ... 99,999,999 Mitte: Erstes von sechs Registern im unten beschriebenen Block Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet Mitte: ein Operand ist ungültig oder außerhalb des Bereichs	Addiert Operand 1 (den Wert des oberen Elements des Registerblocks) und Operand 2 (den Wert in den ersten beidenRegistern des mittleren Elements des Blocks), plaziert dann das Ergebnis im vierten und fünften Register des mittleren Elements des Blocks

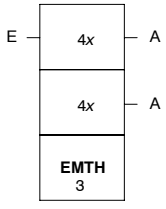
Mittlerer Elemente des Blocks

Registernummer	Registerinhalt
4*x* und 4*x* + 1	Der Wert des Operands 2, Bereich 0 ... 99,999,999
4*x* + 2	ein Wert ungleich Null gibt einen Überlauf an
4*x* + 3 und 4*x* + 4	das Ergebnis der Addition mit doppelter Genauigkeit
4*x* + 5	nicht benutzt, muß jedoch konfiguriert werden

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Doppelte Genauigkeit (32-Bit) Subtraktion		Oben: EIN initiiert die Subtraktion mit doppelter Genauigkeit	Oben: Erstes von zwei aufeinanderfolgenden Registern, die Operand 1 enthalten—sein Wert liegt im Bereich 0 ... 99,999,999 Mitte: Erstes von sechs Registern des unten beschriebenen Blocks Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet Mitte: Operand = Operand 1 2 Unten: Operand < Operand 1 2	Subtrahiert Operand 2 (den Wert in dem ersten und zweiten Register des mittleren Elements des Blocks) von Operand 1 (dem Wert des oberen Elements des Blocks), plaziert dann das Ergebnisim dritten und vierten Registern des mittleren Elements des Blocks

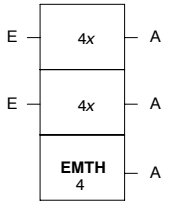
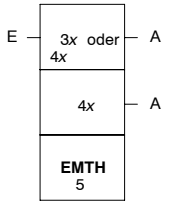
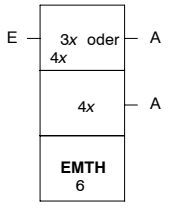
Mittlerer Elemente des Blocks

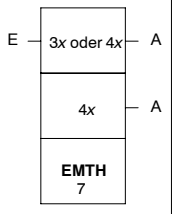
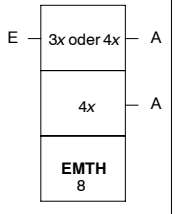
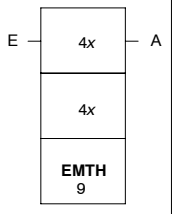
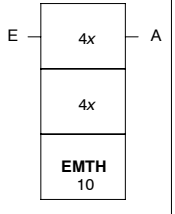
Registernummer	Registerinhalt
4*x* und 4*x* + 1	der Wert des Operand 2, im Bereich 0 ... 99,999,999
4*x* + 2 und 4*x* + 3	das Ergebnis der Subtraktion mit doppelter Genauigkeit
4*x* + 4	ein Wert ungleich Null gibt eine Bereichsüberschreitung an
4*x* + 5	nicht benutzt, muß jedoch konfiguriert werden

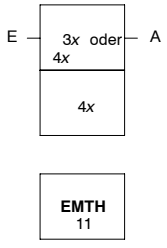
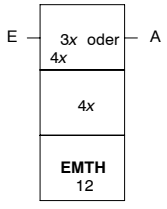
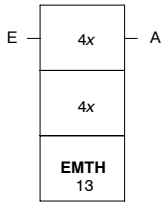
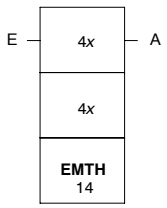
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Multiplikation mit doppelter Genauigkeit		Oben: EIN initiiert die Multiplikation mit doppelter Genauigkeit	Oben: Erstes von zwei aufeinanderfolgenden Registern, das Operand 1 enthält,dessen Wertliegt im Bereich 0 ... 99,999,999 Mitte: Erstes von sechs Registern des unten beschriebenden Blocks Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: ein Operand liegt außerhalb des Bereichs	Multipliziert Operand 1 (den Wert im oberen Element des Registerblocks) mit Operand 2 (dem Wert in den ersten zwei Registern des mittleren Elements des Blocks), plaziert dann das Ergebnis im dritten, vierten, fünften, und sechstem Register des mittleren Elements des Blocks

Mittlerer Elemente des Blocks

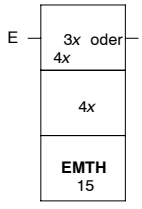
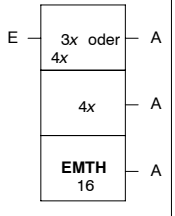
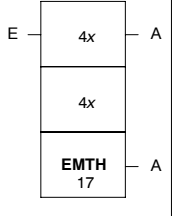
Registernummer	Registerinhalt
4*x* und 4*x* + 1	der Wert von Operand 2, im Bereich 0 ... 99,999,999
4*x* + 2, 4*x* + 3, 4*x* + 4, und 4*x* + 5	das Ergebnis der Multiplikation mit doppelter Genauigkeit

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Division-Doppelte Genauigkeit		Oben: EIN initiiert die Division mit doppelter Genauigkeit Mitte: EIN — Rest wird als Bruch gespeichert AUS — Rest wird als ganze Zahl gespeichert	Oben: Erstes von zwei aufeinanderfolgenden Registern, das Operand 1 enthält—sein Wert liegt im Bereich 0 ... 99,999,999 Mitte: Erstes von sechs Registerndes unten beschriebenen Blocks Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: ein Operand liegt außerhalb des Bereichs Unten: Operand 2 = 0	Dividiert Operand 1 (den Wert im oberen Element des Registerblocks) durch Operand 2 (das erste von zwei Registern im mittleren Element des Blocks), plaziert dann das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks und den Rest in dem fünften und sechsten Registerdes mittleren Elements des Blocks
		Mittlerer Elemente des Blocks			
		Registernummer	**Registerinhalt**		
		4*x* und 4*x* + 1	der Wert von Operand 2, im Bereich 0 ... 99,999,999		
		4*x* + 2 und 4*x* + 3	das Ergebnis (Quotient) der Division mit doppelter Genauigkeit		
		4*x* + 4 und 4*x* + 5	der Rest der Division mit doppelter Genauigkeit		
Quadratwurzel		Oben: EIN initiiert die √ Operation	Oben: Erstes von zwei Registern,das einen Quellwert im Bereich 0 ... 99,999,999 enthält Mitte: Erstes von zwei Registern,wo das Ergebnisin festem Dezimalformat gespeichert wird: 1234.5600 Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet Mitte: Quellwert liegt außerhalb des Bereichs	Berechnet die Quadratwurzel des Quellwerts im oberen Element des Registers und speichert das Ergebnis im mittleren Element des Registers
Prozeß-Quadratwurzel		Oben: EIN initiiert die √ Operation	Oben: Erstes von zwei Registern, das einen Quellwert im Bereich 0 ... 99,999,999 enthält Mitte: Erstes von zwei Registern,wo das linearisierte Ergebnis gespeichert wird Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: Quellwert außerhalb des Bereichs	Berechnet die Quadratwurzel des Quellwerts im oberen Element des Registers, linearisiert diesen durch Multiplizierung mit 63.9922 (der Quadratwurzel von 4095), speichert dann das linearisierte Ergebnis im mittleren Element des Registers. Prozeß Quadrat- wurzeln werden häufig in PID2 Operationen benutzt

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Logarithmus		Oben: EIN initiiert eine logarithmische Operation	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Quellwert im Bereich 0 ... 99,999,999 enthält Mitte: Ein Halteregister, wo das Ergebnis gespeichert wird Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: Ein Fehler wurde festgestellt oder ein Wert liegt außerhalb des Bereichs	Führt eine logarithmische Operation auf der Basis 10 mit den Werten in den Quellegistern des oberen Elements aus, speichert dann das Ergebenis im mittleren Elements des Registers
Anti-Logarithmus		Oben: EIN initiiert eine logarithmische Operation	Oben: Ein einzelnes Register, das einen Quellwert enthält, gespeichert im festen Dezimalformat 1.234 und im Bereich 0 ... 7.999 Mitte: Erstes von zwei aufeinander-folgenden Registern, wo das Ergebnis gespeichert wird Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: ein Fehler wurde festgestellt oder ein Wertliegt außerhalb des Bereichs	Führt eine antilogarithmische Operation mit dem Wert im Quellregister aus und speichert das Ergebnis im mittleren Element des Registers in festem Dezimalformat: 12345678
Integer-zu-Fließpunkt Konversion		Oben: EIN initiiert die Konversion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integer-Quellwert mit doppelter Genauigkeit enthält Mitte: Erstes in einem Block von vier aufeinander- folgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Konvertiert einen Integerwert mit doppelter Genauigkeit in einen 32-Bit Fließpunktwert und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks Das erste von zwei Registern im Block wird nicht benutzt*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks in dem EMTH 9 Befehl konfigurieren.					
Integer + Fließpunkt Addition		Oben: EIN initiiert die Addition	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integerwert mit doppelter Genauigkeit enthält Mitte: erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Addiert den Integerwert mit doppelter Genauigkeit im oberen Elemente des Registerblocks und den FP Wert in den ersten zwei Registern des mittleren Elementes des Blocks, speichert dann das Ergebnis im dritten und vierten Register des mittleren Elementes des Blocks

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Integer - Fließpunkt Subtraktion		Oben: EIN initiiert die Subtraktion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integerwert mit doppelter Genauigkeit enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Subtrahiert den FP Wert in den ersten zwei Registern des mittleren Elements des Blocks von dem Integerwert im oberen Element des Registerblocks, plaziert dann das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks
Integer x Fließpunkt Multiplikation		Oben: EIN initiiert die Multiplikation	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integerwert mit doppelter Genauigkeit enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Registern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Multipliziert den Integerwert mit doppelter Genauigkeit in dem oberen Elemente des Registerblocks mit dem FP Wert in den ersten zwei Registern des mittleren Elements des Blocks, speichert dann das Produkt im dritten und vierten Register des mittleren Elementes des Blocks
Integer/Fließpunkt Division		Oben: EIN initiiert die Division	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integerwert mit doppelter Genauigkeit enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Dividiert den Integerwert mit doppelter Genauigkeitim oberen Elemente desRegisterblocks durch den FP Wert in den ersten zwei Registern des mittleren Elementeblocks, speichert dann den Quotient im dritten und vierten Register des mittleren Elementes des Blocks
Fließpunkt - Integer Subtraktion		Oben: EIN initiiert die Subtraktion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Fließpunktwert enthält Mitte: Erstes in einem Block von vier aufeinander- folgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Subtrahiert den Integerwert mit doppelter Genauigkeit in den ersten zwei Registern des mittleren Elements des Blocks von dem FP Wert im oberen Element des Registerblocks, speichert dann das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks

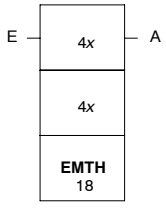
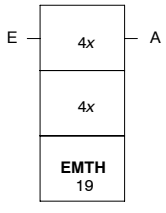
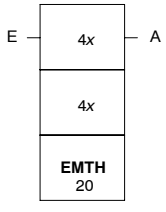
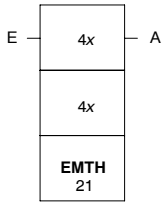
Erweiterte mathematische Befehle (Fortsetzung)

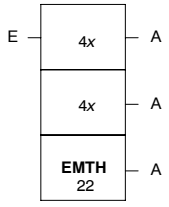
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt/ Integer Division		Oben: EIN initiiert die Division	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Fließpunktwert enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Dividiert den Integerwertmit doppelter Genauigkeitin den ersten zwei Registern des mittleren Elements des Blocks durch den FP Wert im oberen Element des Registerblocks, speichert dann den Quotient im dritten und vierten Register des mittleren Elements des Blocks
Integer-Fließpunkt Vergleich		Oben: EIN initiiert den Vergleich	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen Integerwertmit doppelter Genauigkeitenthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: mit dem unteren Ausgang benutzt, um das Wertverhältnis anzugeben Unten: mit dem mittleren Ausgang benutzt, um das Wertverhältnis anzugeben	Vergleicht den Integerwert mit doppelter Genauigkeit mit dem Fließpunkt Wert (in den ersten zwei Registern des mittleren Elements des Blocks), gibt dann das Verhältnisüber diemittleren und unteren Ausgänge an (vgl.untenstehende Tabelle) Das dritte und vierte Register im mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden
Fließpunkt-zu-Integer Konversion		Oben: EIN initiiert die Konversion	Oben: Erstes von zwei aufeinander folgenden Registern, das einen Integerwertmit doppelter Genauigkeit enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN wenn Berechnung beendet Unten: 0 = + Integerwert 1 = - Integerwert	Konvertiert den FP Wert, der im dritten und vierten Register des mittleren Elements des Blocks gespeichert ist in einen Integerwert mit doppelter Genauigkeit und speichert den konvertierten Wert im oberen Elemente des Registers. Das erste und zweite Register im mittleren Element werden nicht benutzt, müssen aber konfiguriert werden*

EMTH 16 Ausgänge

Ausgangsstatus Mitte	Ausgangsstatus Unten	Wertverhältnis
EIN	AUS	I > FP
AUS	EIN	I < FP
EIN	EIN	I = FP

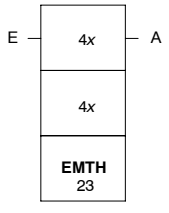
*** Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Interwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elemente des Blocks speichern und keinen oberen Elemente des Registerblocks des EMTH 17 Befehls konfigurieren.

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt Addition		Oben: EIN initiiert die Subtraktion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen FP Wert 1 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Addiert FP Wert 1 (im oberen Element des Registerblocks) und FP Wert 2 (aus den ersten zwei Registern des mittleren Elements des Blocks), speichert dann die Summe im dritten und vierten Register des mittleren Elements des Blocks
Fließpunkt Subtraktion		Oben: EIN initiiert die Multiplikation	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen FP Wert 1 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Subtrahiert FP Wert 2 (gespeichert im ersten und zweiten Register des mittleren Elements des Blocks) von FP Wert 1 (in oberen Element des Registerblocks), speichert dann die Differenz im dritten und vierten Register des mittleren Elements des Blocks
Fließpunkt Multiplikation		Oben: EIN initiiert die Division	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert 1 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Multipliziert FP Wert 1 (im oberen Element desRegisterblocks) mit FP Wert 2 (gespeichert im ersten und zweiten Register des mittleren Elementeblocks), speichert dann das Produkt im dritten und vierten Register des mittleren Elements des Blocks
Fließpunkt Division		Oben: EIN initiiert die Subtraktion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert 1 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Dividiert FP Wert 1 (im oberen Element des Registerblocks) durch FP Wert 2 (gespeichert im ersten und zweiten Register des mittleren Elementeblocks), speichert dann den Quotient im dritten und vierten Register des mittleren Elements des Blocks

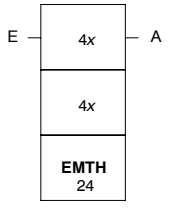
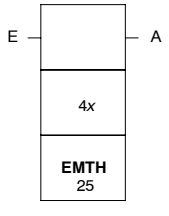
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt Vergleich		Oben: EIN initiiert den Vergleich	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert 1 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionskode	Oben: EIN, wenn Vergleich beendet Mitte: mit dem unteren Ausgang benutzt, um das Wertverhältnis anzugeben Unten: mit dem mittleren Ausgang benutzt, um das Wertverhältnis anzugeben	Vergleicht FP Wert 1 (im oberen Element des Registerblocks) und FP Wert 2 (in den ersten zwei Registern des mittleren Elements des Blocks), gibt dann das Verhältnismittels der mittleren und unteren Ausgänge an (vgl. untenstehende Tabelle) Das dritte und vierte Register im mittleren Element des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden

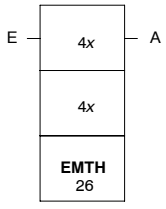
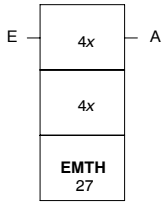
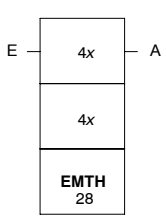
EMTH 22 Ausgänge

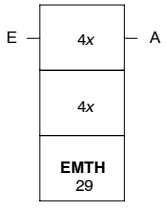
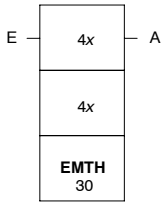
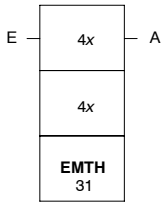
Ausgangsstatus Mitte	Ausgangsstatus Unten	Wertverhältnis
EIN	AUS	FP Wert 1 > FP Wert 2
AUS	EIN	FP Wert 1 < FP Wert 2
EIN	EIN	FP Wert 1 = FP Wert 2

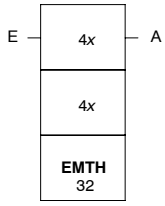
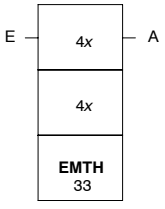
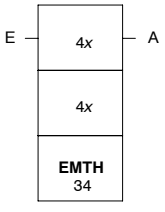
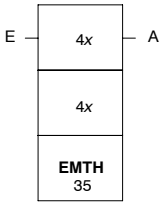
Fließpunkt Quadratwurzel		Oben: EIN initiiert die √ Operation	Oben: Erstes von zwei aufeinanderfolgenden Registern, das einen FP Wert enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Führt eine Quadratwurzel Operation mit dem FP Wert im oberen Element des Blocks aus und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweiteRegister im mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden*

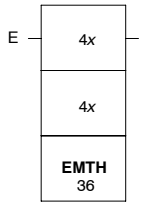
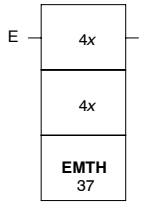
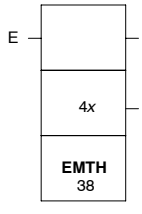
*** Hinweis** Wenn Sie Register aufbewahren möchten, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks im EMTH 23 Befehl konfigurieren.

Fließpunkt Vorzeichenwechsel		Oben: EIN initiiert die Vorzeichenwechsel Operation	Oben: Erstes von zwei Registern, das einen FP Wert enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Operation beendet	Ändert das Vorzeichen des FP Werts im oberen Element des Registerblocks und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt
Fließpunkt π laden		Oben: EIN lädt π in den mittleren Registerblock	Oben: Nicht benutzt Mitte: Erstes von vier Registern, bei denen einFP Wert von pi geladen wird Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Laden beendet	Lädt einen FP Wert von pi in das dritte und vierte Register des mittleren Elements des Blocks; das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt

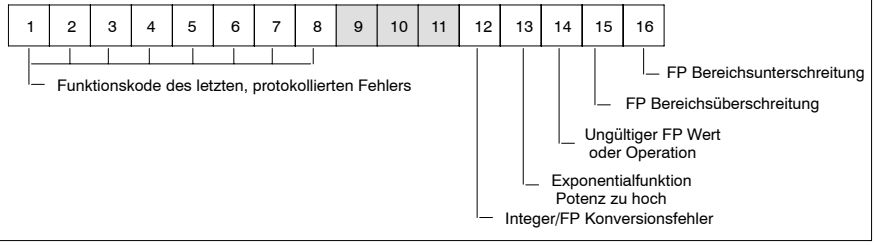
Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt Sinus eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert eines Winkels in Bogenangaben enthält; der Größenbereich ist < 65536.0 Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Berechnet in Bogenangaben den Sinus des Fließpunkt-werts im oberen Elemente des Registers und speichert das Ergebnis im dritten und vierten Register des mittleren Elemente des Blocks. Das erste und zweite Register des mittleren Elemente des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elemente des Blocks speichern und keinen oberen Elemente des Registerblocks in dem EMTH 26 Befehl konfigurieren.					
Fließpunkt Kosinus eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert eines Winkels in Bogenangaben enthält; der Größenbereich ist < 65536.0 Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Berechnet in Bogenangaben den Kosinus des Fließpunkt-Werts im oberen Elemente des Registers und speichert das Ergebnis im dritten und viertenRegister des mittleren Elemente des Blocks. Das erste und zweite Register des mittleren Elemente des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Elemente des Registerblocks in dem EMTH 27 Befehl konfigurieren.					
Fließpunkt Tangente eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert eines Winkels in Bogenangaben enthält; der Größenbereich < 65536.0 Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Berechnet in Bogenangaben die Tangente des Fließpunkt-Werts im oberen Element des Registers und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt,müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks in dem EMTH 28 Befehl konfigurieren.					

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt Arcussinus eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei Regsistern,das den FP Wert des Sinus eines Winkels zwischen -π / 2 ... π / 2 Bogenangaben enthält; der Wert muß im Bereich -1.0 ... +1.0 liegen Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Berechnet in Bogenangaben den Arcussinus des Fließpunkt Wertes im oberen Element desRegisters und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks;. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks in dem EMTH 29 Befehl konfigurieren.					
Fließpunkt Arcuscosinus eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei-Registern, das den FP Wert des Kosinus eines Winkels zwischen 0 ... π Bogenangaben enthält; im Bereich von -1.0 ... +1.0 Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Berechnet in Bogenangaben den Arcuscosinus Fließpunkt Wert im oberen Element des Registers und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweite Register im mittleren Element des Blocks werden nicht benutzt,müssen jedoch konfiguriert werden*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks in dem EMTH 30 Befehl konfigurieren.					
Fließpunkt Arcustangens eines Winkels		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinander-folgenden Register, das den FP Wertder Tangente eines Winkels zwischen -π/2 ... π/2 Bogenangaben Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Berechnet in Bogenangaben den Arcustangens des Fließpunkt-Wertes im oberen Element desRegisters und speichert das Ergebnis im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im ersten und zweiten Register des mittleren Elements des Blocks speichern und kein oberes Element des Registerblocks in dem EMTH 31 Befehl konfigurieren.					

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt Bogenangabe-zu-Grad Konversion		Oben: EIN initiiert die Konversion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert eines Winkels in Bogenangaben enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Konversionbeendet	Konvertiert den FP Wert im oberen Elemente des Registers zu einer FP Darstellung dieses Werts in Bogenangaben, und speichert die Konversion im dritten und vierten Register des mittleren Element des Blocks. Das erste und zweite Register des mittleren Elemente des Blocks werden nicht benutzt,müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im 1. und 2. Register des mittleren Elements des Blocks speichern und keine oberen Elemente des Registerblocks in dem EMTH 32 Befehl konfigurieren.					
Fließpunkt Grad-zu-Bogenangaben Konversion		Oben: EIN initiiert die Konversion	Oben: Erstes von zwei aufeinanderfolgenden Registern, das den FP Wert eines Winkels in Grad enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Konversion beendet	Konvertiert den FP Wert im oberen Elemente des Registers zu FP Darstellung dieses Werts in Grad, und speichert den konvertierten Wert im dritten und vierten Register des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt,müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im 1. und 2. Register des mittleren Elements des Blocks speichern und keine oberen Elemente des Registerblocks in dem EMTH 33 Befehl konfigurieren.					
Fließpunkt Zahl, zu einer Integerpotenz erhoben		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei Registern, das einen FP Wert enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Erhebt den FP Wert im oberen Element desRegisters zu der Integerpotenz, die im zweiten Register des mittleren Elements des Blocks angegeben ist, und speichert das Ergebnis im dritten und viertenRegister des mittleren Elements des Blocks. Das erste Register im mittleren Element muß auf Null gesetzt sein
Fließpunkt exponential		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinanderfolgenden Registern, das 1 FP Wert im Bereich -87.34 ... +88.72 enthält Mitte: Erstes in einem Block von 4 aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnungbeendet	Berechnet den Exponentialwert der FP Zahl im oberen Elemente des Registers und speichert das Ergebnis im dritten und viertenRegister des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im 1. und 2. Register des mittleren Elements des Blocks speichern und keine oberen Elemente des Registerblocks in dem EMTH 35 Befehl konfigurieren.					

Befehl	Struktur	Eingänge (E)	Elemente	Ausgänge (A)	Funktion
Fließpunkt natürlicher Logarithmus		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinander-folgenden Registern, das einen FP Wert > 0 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Berechnet den natürlichen Logarithmus des FP Werts im oberen Element des Registers und speichert das Ergebnis im dritten und viertenRegister des mittleren Elements des Blocks Das erste und zweite Register des mittleren Elements des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im 1. und 2. Register des mittleren Elemente des Blocks speichern und keinen oberen Elemente des Registerblocks in dem EMTH 36 Befehl konfigurieren.					
Fließpunkt Logarithmus zur Basis 10		Oben: EIN initiiert die Berechnung	Oben: Erstes von zwei aufeinander-folgenden Registern, das einen FP Wert > 0 enthält Mitte: Erstes in einem Block von vier aufeinanderfolgenden Halteregistern Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet	Berechnet den Logarithmuszur Basis 10 der FP Zahl im oberen Elemente des Registers und speichert das Ergebnis im dritten und viertenRegister des mittleren Elements des Blocks. Das erste und zweite Register des mittleren Elemente des Blocks werden nicht benutzt,müssen jedoch konfiguriert werden.*
* **Hinweis** Wenn Sie Register aufbewahren wollen, können Sie den Integerwert mit doppelter Genauigkeit im 1. und 2. Register des mittleren Elemente des Blocks speichern und keinen oberen Elemente des Registerblocks in dem EMTH 37 Befehl konfigurieren.					
Fehlermeldung Protokoll		Oben: EIN initiiert die Berechnung	*Oben:* Nicht benutzt Mitte: Erstes von vier-Register, die die Fehlerprotokoll-Daten enthalten (vgl. unten) Unten: entsprechender EMTH Funktionscode	Oben: EIN, wenn Berechnung beendet Mitte: 1 = Nicht-Nullen im Register 0 = Alle Bits auf Null gesetzt	Fehlerdaten werden im dritten Register des mittleren Elements des Blocks protokolliert, und das vierte Register wird immer auf Null gesetzt Das erste und zweite Register im mittleren Element des Blocks werden nicht benutzt, müssen jedoch konfiguriert werden.

Register 4*x* + 2 im mittleren Elemente von EMTH 38



Anhang A
Update des Betriebssystems im Flash

- ☐Dienstprogramme für Betriebssystem-Updates
- ☐Zugriff auf Modfax

Dienstprogramme für Betriebssystem-Updates

Fabrikmäßig wurde das Kontaktplanlogik-Betriebssystem Ihrer Modicon Micro SPS in den Flashspeicher der SPS geladen. Das Betriebssystem definiert die Grundfunktionen der SPS und ihrem Befehlsvorrat.

Die Durchführung von Updates kann notwendig werden, um die Systemfunktionen zu erweitern oder Fehler zu beheben. Die folgende Information soll Ihnen helfen, falls Sie ein Update des Flash durchführen müssen.

Update des Systems mit einem Kontaktplanlogik-Programm

Obwohl der Flashspeicher nicht-flüchtig ist, kann er leicht geändert werden. Sie können ein Update mit einer neueren Betriebssystemversion über den Modbus-Port, d.h. ohne irgendwelche Veränderungen an der Hardware, ausführen.

Erforderlich ist lediglich die Datei mit der binären Betriebssystem-Software und ein Dienstprogramm zum Laden. Sie können die binäre Betriebssytemdatei von einem PC aus in die Micro SPS mit dem Laden-Dienstprogramm übernehmen.

Das Laden-Dienstprogramm enthält fünf Dateien:

- LOADER.EXE, eine ausführbare Datei, die die Ladefunktionen durchführt
- LOADER.HLP, eine Datei mit Hilfe-Text
- LOADER.NDX, eine Indexdatei für Hilfe-Bildschirme
- MCMIII.MSG, die Nachrichtendatei mit Modcom III Fehlern
- README.1ST ist eine Datei, die genau erklärt, wie das Update durchgeführt wird

Das Laden-Dienstprogramm und die neueste Betriebsystemsoftware können angefordert werden bei:

- dem Customer Service Bulletin Board (24 Stunden täglich, 365 Tage im Jahr, kostenlos)
- Von Ihrem nächsten Modicon-Händler

Die neuesten Versionen der verschiedenen Betriebssysteme (oder Betriebssystem-Firmware) sind im Customer Service BBS und im Modfax aufgelistet. Die Modfax-Dokumentennummer für die neuesten Micro SPS Upgrades ist 3727. Einzelheiten über den Zugang zu Modfax und zum Customer Service BBS finden Sie in diesem Anhang.

Feststellen der neuesten, verfügbaren Version

Es gibt zwei Möglichkeiten festzustellen, welche Version des Betriebssystems gegenwärtig in einer SPS installiert ist.

- ☐ Wenn Sie MODSOFT Lite besitzen, können Sie die Exec ID Zeile des Status-Bildschirms der Steuerung prüfen — wenn z.B. der Informationsbildschirm des Status der SPS anzeigt:
  00
  ```
  EXEC ID 0861 REV 0101
  ```
  dann ist die Versionsnummer des Betriebssystems 1.01.
- ☐ Wenn Sie nicht über MODSOFT oder MODSOFT Lite verfügen, können Sie die folgenden, absoluten Speicheradressen aufrufen, um sich die Version des SPS-Betriebssystems anzeigen zu lassen:

Steuerung	Seite	Adresse
Alle Micro SPS	F	4020 (hex)

Der Zugriff zu den obigen Adressen hängt von der Software ab, die Sie benutzen. Für Details nehmen Sie bitte Kontakt mit Ihrem Software-Lieferanten auf.

Zugang zu Modfax

Modfax ist ein automatisches Dokumenten-Abfragesystem, das den Modicon Kunden zur Verfügung steht. Das System ist selbst-abfragend. Rufen Sie für die Herstellung einer Modfax-Verbindung die Telefonnummer (800) 468-5342 an und wählen Sie Option 3. Halten Sie beim Anruf Ihre Faxnummer bereit.

Zusätzliche technische Hilfe für Hardware oder Software erhalten Sie vom Modicon Field Support Center über (800) 468-5342 oder (508) 794-0800 (außerhalb der U.S.A. und Canadas), wählen Sie dafür Option 1.

Zugang zum Customer Service Bulletin Board

Das Modicon Customer Service BBS enthält mehrere Funktionen und bietet eine Reihe von Vorteilen. Für weitere Information sollten Sie das Modfax Dokument #1113 anfordern oder sich an das Modicon Field Support Center wenden.

BBS-Mitglieder können das unten aufgeführte Verfahren benutzten, oder direkt zur `Flash Lib` gehen. Das Herunterladen des Flash-Betriebssystems kostet keinerlei Gebühren.
Nicht-Mitglieder von BBS sollten das folgende Verfahren wählen, um die binäre Betriebssytemdatei und das Laden-Dienstprogramm im BBS zu finden:

Betriebssystem Update-Prozedur

Schritt1. Wählen Sie mit Hilfe Ihres Modems und Kommunikations-Paketes die Nummer 508-975-9779. Benutzen Sie die höchste Baudrate Ihres Modems — Wir unterstützen bis zu 14,400 Baud, keine Parität, 8 Datenbits und 1 Stopbit.

Schritt2. Wenn es Ihr erster Anruf ist, müssen Sie zunächst ein Konto eröffnen – Beantworten Sie die fünf Fragen, die Ihnen zu diesem Zweck gestellt werden.

Schritt3. Wenn Sie zum Hauptmenü kommen, wählen Sie bitte `m` und betätigen Sie <**Enter**>. Sie werden zum Flash Download-Service begrüßt.

Schritt4. Das Menü zeigt eine Anzahl von SPS-Modellen. Wählen Sie die Zahl, die Ihrem Modell entspricht.

Schritt5. Sie erhalten eine Liste mit Dateien, von 1 ... 8 numeriert, mit einer Beschreibung jeder Datei auf der rechten Seite des Bildschirms. Wählen Sie die Nummer mit der neuesten Version Ihrer SPS — normalerweise 1 oder 2.

Schritt6. Wählen Sie die Protokolle zum Laden, die dem Protokoll Ihres Kommunikationspakets entsprechen. Benutzen Sie ZMODEM, wenn Sie darüber verfügen — anderenfalls KERMIT oder XMODEM.

Schritt7. Wenn Ihr Paket ZMODEM besitzt, wird das Laden automatisch fortgesetzt. Bei anderen Protokollen müssen Sie ggf. Ihrer Kommunikationssoftware mitteilen,daß Sie eine Datei laden wollen, dann das entsprechende Protoll wählen, das dem vorher im BBS gewählten entspricht.

Schritt8. Jetzt sollte sich die entsprechende Datei in Ihrem Pfad befinden (festgelegt durch das entsprechende Kommunikationspaket).

Schritt9. Nehmen wir an, daß die erste von Ihnen übernommene Datei das binäre Betriebssystem ist. Als nächstes möchten Sie das Laden-Dienstprogramm abrufen. Betätigen Sie <**Enter**> einmal — dadurch werden Sie zum Hauptmenü zurückgeführt. Falls dies nicht geschieht, geben Sie `/GO EXEC` ein und betätigen Sie <**Enter**>.

Schritt10. Wiederholen Sie die obige Prozedur, um das Laden Dienstprogramm zu übernehmen. Beginnen Sie dafür dieses Mal bei **Schritt5** mit dem Buchstaben `L`.

Schritt11. Die geladenen Dateien sind komprimiert und entpacken sich selbst, wenn sie ausgeführt werden.

Wenn eine geladene binäre.exe Betriebssystemdatei gestartet wird, entsteht eine binäre Betriebssystemdatei.

Schritt12. Folgen Sie den Anleitungen, die in der README.1ST Datei für das Update des Kontaktplanlogik-Betriebssystems gegeben werden.

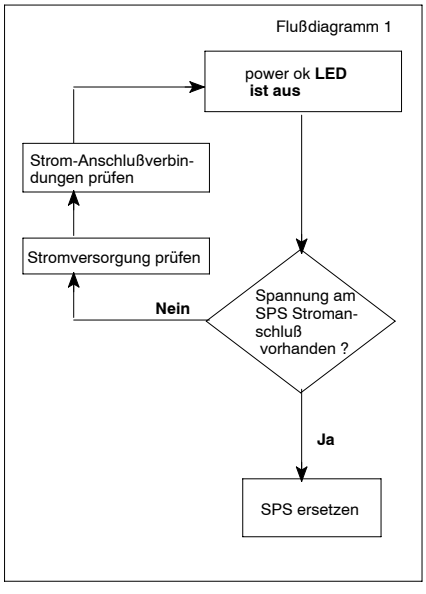
Anhang B
Fehlerbeseitigung

- ☐Diagnose der Startbedingungen
- ☐Fehlercodes bei SPS-Stop
- ☐Auf den LEDs angezeigte SPS-Crashcodes

Diagnose der Startbedingungen

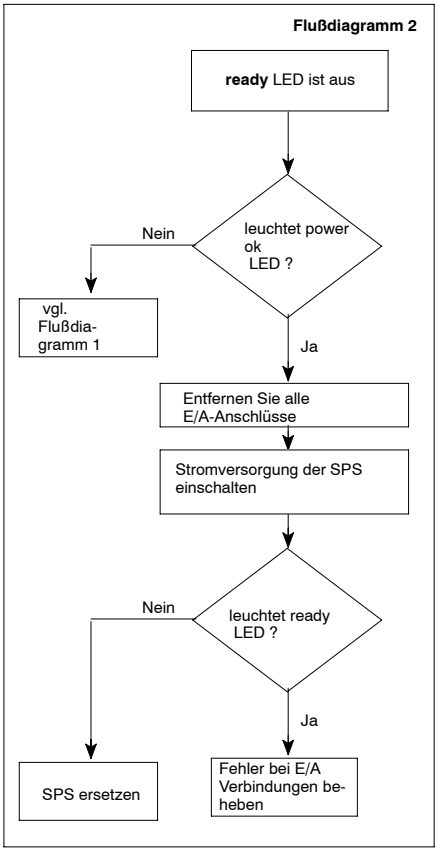
Symptom: *powerok* LED leuchtet nicht

Die grüne LED **power ok** der Micro SPS leuchtet, wenn die interne Stromversorgung der SPS in Ordnung ist und die SPS von einem externen Netzteil Strom erhält. Vergleichen Sie Flußdiagramm 1,falls diese LED nach Einschalten des Stroms der SPS nicht leuchtet.



Symptom: *ready*LED leuchtet nicht

Die gelbe LED **ready** leuchtet, sobald die SPS erfolgreich die Start-Diagnosetests durchlaufen hat und bleibt an, solange die SPS Strom hat und in Ordnung ist. Vergleichen Sie Flußdiagramm 2, falls diese LED nach dem Einschalten nicht leuchtet.



Symptom: *run* LED leuchtet nicht oder blinkt

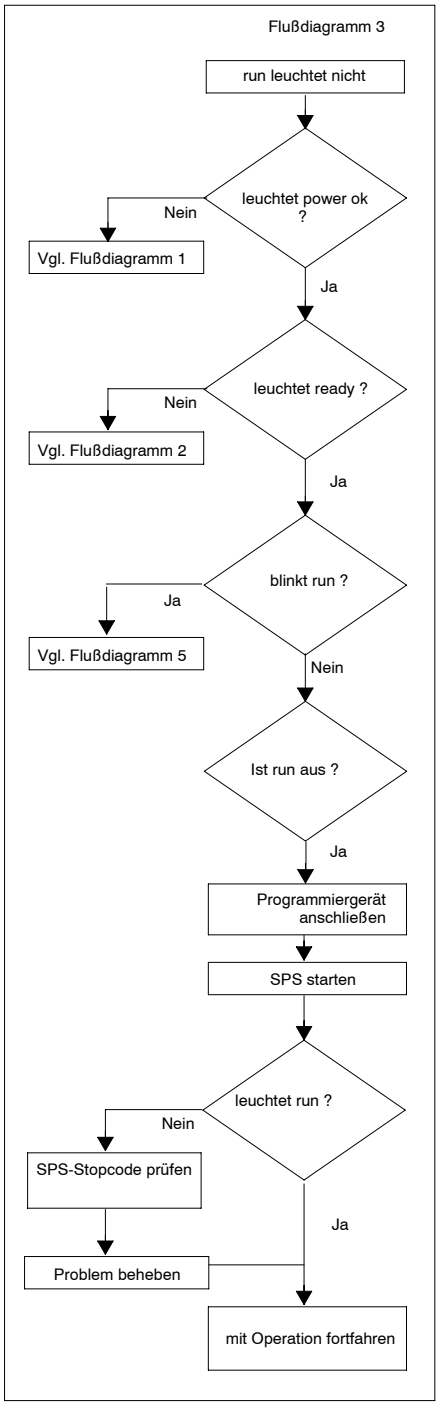
Die LED **run** an der SPS leuchtet permanent, wenn die SPS gestartet wurde und Kontaktplanlogik bearbeitet. Sie blinkt, wenn die SPS Strom hat, aber keine gültige Konfiguration finden kann. Vergleichen Sie Flußdiagramm 3 auf der nächsten Seite, falls die LED aus ist oder sich ungewöhnlich verhält.

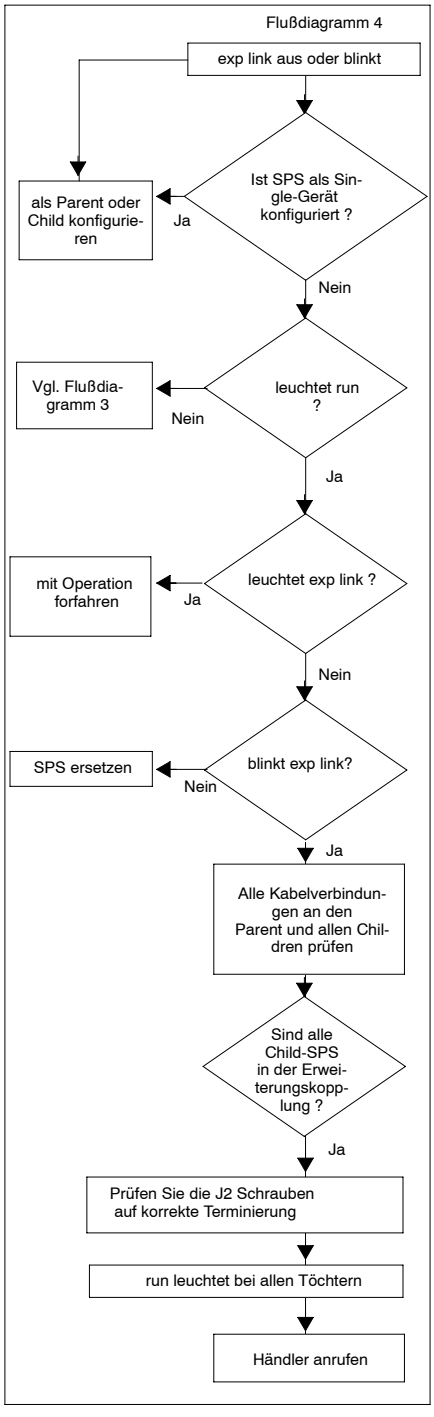
Symptom: *exp link* LED leuchtet nicht oder blinkt

Die grüne LED **exp link** leuchtet permanent, wenn gültige Kommunikationen auf der E/A-Erweiterungskopplung zwischen einer Parent und Child-SPS stattfinden. Die LEDs auf beiden SPS müßten sich gleich verhalten. Bei einem Blinken liegt ein Fehler in der Kopplung vor. Vergleichen Sie Flußdiagramm 4 auf der nächsten Seite, falls Sie die LED blinken sehen oder sie aus ist, wenn eigentlich eine E/A-Kommunikation stattfinden müßte.

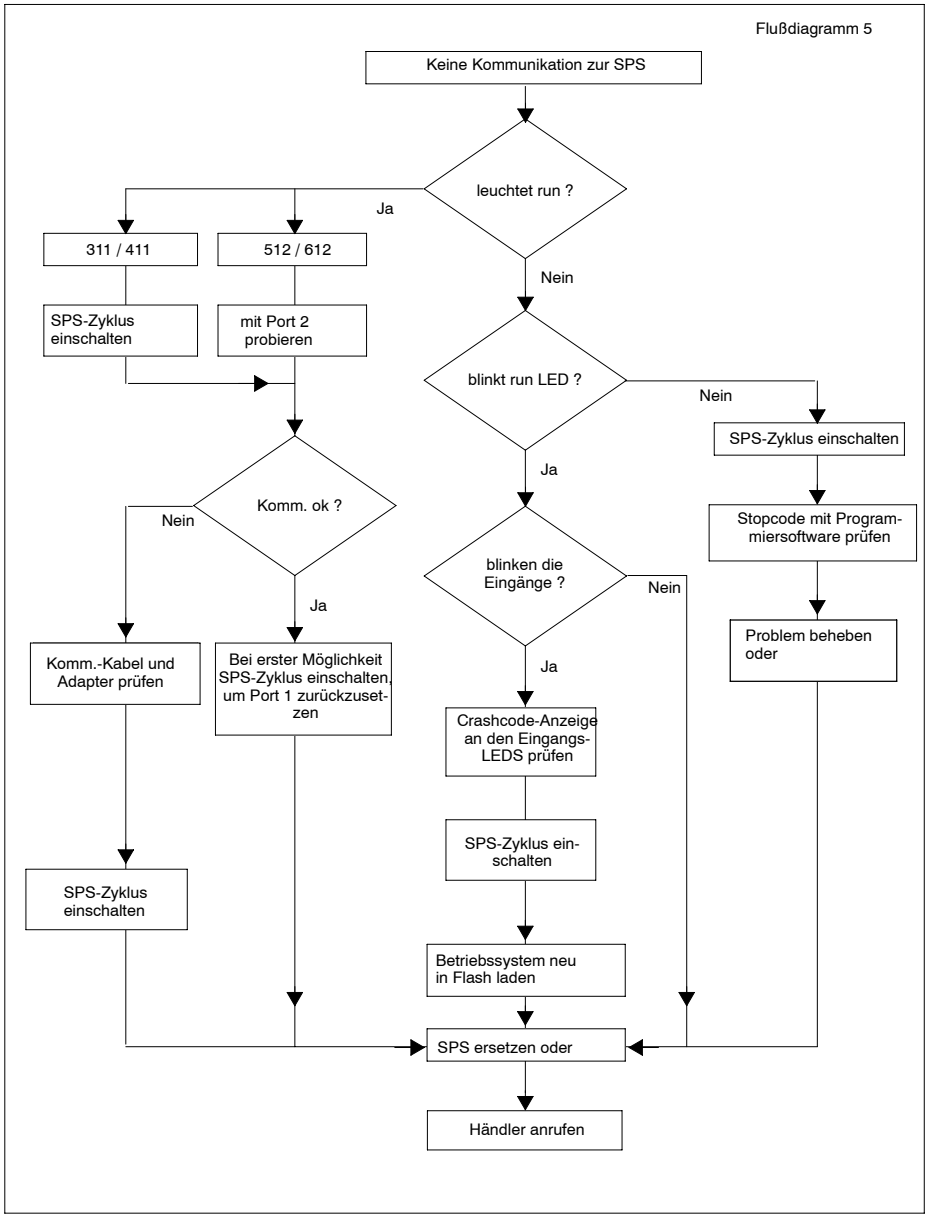
Symptom: Keine Kommunikation zur SPS

Vergleichen Sie Flußdiagramm 5, falls die Kommunikation zur SPS unerwartet ausfällt. Sie sollten dabei auch prüfen, ob ein SPS-Stop Fehlercode auf dem Display Ihres Programmiergeräts angezeigt wird oder eine Crashcode-Anzeige an den Eingangs-LEDs der Micro SPS aufblinkt. Die SPS-Stop Fehlercodes und die System-Crashcodes werden weiter hinten in diesem Anhang beschrieben.


Flußdiagramm 3
run leuchtet nicht
leuchtet power ok ?
Nein
Vgl. Flußdiagramm 1
Ja
leuchtet ready ?
Nein
Vgl. Flußdiagramm 2
Ja
blinkt run ?
Ja
Vgl. Flußdiagramm 5
Nein
Ist run aus ?
Ja
Programmiergerät anschließen
SPS starten
leuchtet run ?
Nein
SPS-Stopcode prüfen
Problem beheben
Ja
mit Operation fortfahren


Flußdiagramm 4
exp link aus oder blinkt
Ist SPS als Single-Gerät konfiguriert ?
Ja
als Parent oder Child konfigurieren
Nein
leuchtet run ?
Nein
Vgl. Flußdiagramm 3
Ja
leuchtet exp link ?
Ja
mit Operation forfahren
Nein
blinkt exp link?
Nein
SPS ersetzen
Ja
Alle Kabelverbindungen an den Parent und allen Children prüfen
Sind alle Child-SPS in der Erweiterungskopplung ?
Ja
Prüfen Sie die J2 Schrauben auf korrekte Terminierung
run leuchtet bei allen Töchtern
Händler anrufen

043502910


Flußdiagramm 5
Keine Kommunikation zur SPS
leuchtet run ?
Ja
Nein
311 / 411
512 / 612
SPS-Zyklus einschalten
mit Port 2 probieren
blinkt run LED ?
Nein
SPS-Zyklus einschalten
Komm. ok ?
Nein
Ja
Stopcode mit Programmiersoftware prüfen
blinken die Eingänge ?
Nein
Problem beheben oder
Komm.-Kabel und Adapter prüfen
Bei erster Möglichkeit SPS-Zyklus einschalten, um Port 1 zurückzusetzen
Ja
Crashcode-Anzeige an den Eingangs-LEDS prüfen
SPS-Zyklus einschalten
SPS-Zyklus einschalten
Betriebssystem neu in Flash laden
SPS ersetzen oder
Händler anrufen

SPS Stopfehler-Codes

Wenn eine SPS unerwartet stoppt, wird ein Stop-Fehlercode mit Hilfe der Programmiergerätsoftware angezeigt. Der Code wird als vier Zeichen lange Hexadezimalzahl angegeben. Bei MODSOFT Lite wird der Stop-Fehlercode auf dem SPS-Statusbildschirm gezeigt,beim HHP wird der Stop-Fehlercode ??? gezeigt

Die Bedeutung der verschiedenen Codes sind in untenstehender Tabelle aufgelistet.

Stop-Fehlercode		
Stop-Bitcode	**Stop-Ursache**	**Beschreibung**
8000	PCSTOPPED	SPS ist gestoppt
4000	BADTCOP	Fehler im E/A-Verzeichnis
2000	DIMAWAR	SPS hat keine gültige Konfiguration
1000	PORTIVENT	Port-Eingriffsfehler
0800	BADSEGSCH	Kontaktplanlogik-Segmente sind für Zyklus nicht korrekt angeordnet
0400	SONNEINTIST	Das Netzwerk-Start Element hat das Netzwerk nicht gestartet
0200	PDCHECKSUM	Checksum Diagnose beim Abschalten fehlerhaft
0080	NEINEOLDOIO	Watchdog-Timer abgelaufen, bevor Zyklus beendet
0040	RTCFAILED	Fehler bei Echtzeit-Uhr
0020	BADOXUSED	Fehler in der Tabelle der benutzten Spulen
0010	RIOFAILED	Fehler in der E/A-Erweiterungskopplung
0008	NODETYPE	Es wurde ein illegales Element benutzt
0004	ULCSUMERR	Checksum-Fehler in der Anwenderlogik
0002	DSCRDISAB	Fehler beim Disablen eines diskreten Elements
0001	BADCONFIG	Fehler in der Konfiguartion

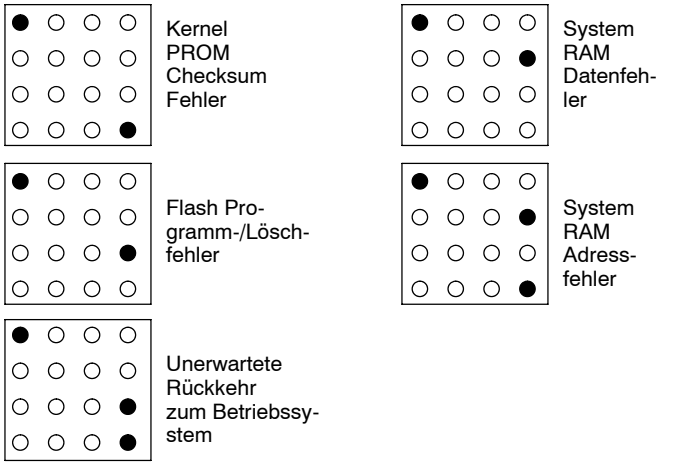
An den LEDs angezeigte SPS-Crashcodes

Wenn die CPU einen Fehler feststellt, wird einer der unten aufgeführten Fehlercodes an den Eingangs-LEDS auf der Vorderseite der SPS blinken. Die LED **ready** wird kontinuierlich leuchten und die **run** LED wird mit der gleichen Frequenz wie die Eingangs-LEDs blinken — 0.5 s ein und 2.5 s aus.

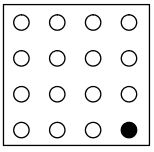
Es gibt zwei Kategorien der SPS-Crashcodes, solche, die von der SPS im*Kernel*-Modus erzeugt werden, und solche, die während der Anwendung entstehen.

In der untenstehenden Abbildung werden blinkende Eingangs-LEDs als ● dargestellt und Eingangs-LEDS, die aus sind, als ○ .

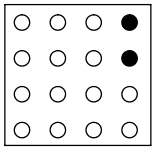
Kernel-Modus Crashcodes



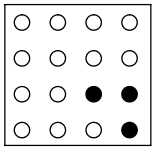
Anwendungs-Crashcodes



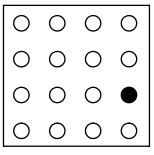
PROM Checksum Fehler



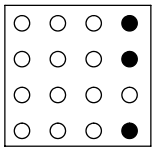
Run Fehler bei Ausgangsaktivierung



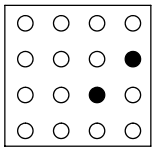
Fehler Comm-Status —beim Senden von RTU



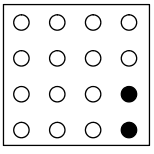
RAM Datentest-Fehler



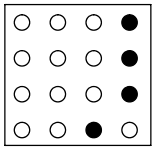
Unerwarteter Interrupt



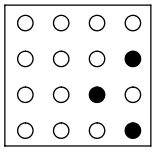
Fehler Comm-Status —beim Empfang von RTU



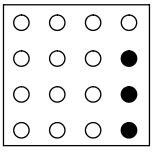
RAM Adressentest-Fehler



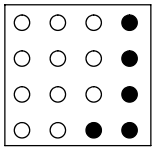
Fehler UART Hardware



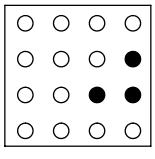
Fehler Comm-Status —beim Empfang vonASCII



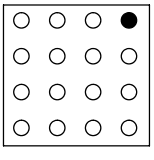
Modbus-Befehl Pufferüberlauf



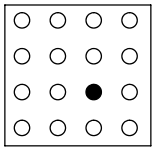
Fehler UART Interrupt (extern zum Microprozessor)



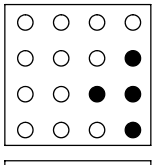
Fehler Modbus-Status —Timer 0 Ereignis



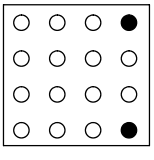
Modbus-Befehl Länge = 0



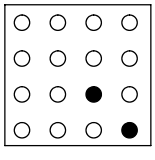
Fehler Empfang Comm-Status



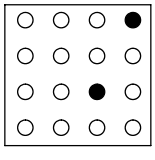
Fehler Modbus Status—Senden von Interrupt



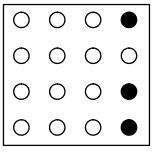
Modbus Abbruch-Befehlsfehler



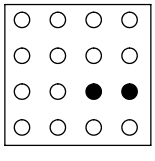
Fehler beim Senden Comm-Status



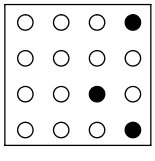
Fehler Modbus- Status — empfangener Interrupt



RAM-Fehler bei Dimensionierung



Fehler Comm-Status—beim Senden von ASCII



Zeitabschaltfehler beim-Herstellen von ready